Massively Parallel Bayesian Inference Framework for GPU Supercomputers – Application to Estimation of Coseismic Fault Slip –

Kai Nakao
k-nakao@eri.u-tokyo.ac.jp
Earthquake Research Institute,
The University of Tokyo
Tokyo, Japan

Tsuyoshi Ichimura
ichimura@eri.u-tokyo.ac.jp
Earthquake Research Institute,
The University of Tokyo
Tokyo, Japan

Kohei Fujita
fujita@eri.u-tokyo.ac.jp
Earthquake Research Institute,
The University of Tokyo
Tokyo, Japan

1 Introduction

This research presents a massively parallel Bayesian inference framework for GPU-based supercomputers. Inverse analysis, which estimates unobservable parameters from observable data, is a key process in many scientific and engineering fields. Bayesian inference provides a robust approach by combining data with prior knowledge and quantifying uncertainty. However, practical applications often rely on Monte Carlo sampling, which requires forward simulations for a large number of parameter samples (often exceeding 100,000), leading to the need for massively parallel computing solutions. A state-of-the-art massively parallel Bayesian inference implementation, targeted to the estimation of coseismic fault slip[5], was developed for CPU-based supercomputers. It suffers from long time-to-solution (e.g., 3.5 hours on 1024 nodes of Fugaku[9]) and is unsuitable for GPU acceleration because its workload consists of many small, imbalanced computations. In this study, we developed a new methodology designed for GPU architectures. This approach reduces the time-to-solution to just 39.4 minutes on 128 nodes of the GPU-based supercomputer Miyabi[10], corresponding to a 42.1-fold speedup per node and reducing the energy-to-solution to 18.8% compared to the original CPU implementation.

2 Methodology and Performance

This study addresses Bayesian estimation of coseismic fault slip, simultaneously inferring fault geometry θ and slip distribution s from crustal deformation data d. The computational core is hierarchical Monte Carlo sampling. The goal is to generate samples from the posterior PDF $P(\theta \mid d)$, while the likelihood $P(d \mid \theta)$ is computed by integrating over s. Sequential Monte Carlo (SMC), a sampling algorithm well suited for parallelization, is employed in a nested manner[3]: the outer SMC loop samples θ parallelized across multiple nodes, and the inner SMC loop samples s with θ fixed which runs on GPUs within each node to evaluate $P(d \mid \theta)$ via Monte Carlo integration.

In this study, the most computationally demanding part of the algorithm, Hamiltonian Monte Carlo (HMC)[4] in the inner SMC for s, is offloaded to GPUs. The original CPU implementation following Ching & Chen (2007)[1] results in imbalanced workloads across threads because the number of HMC steps varies for each sample s_i. For the GPU implementation, we adopt another SMC implementation that enforces a uniform number of HMC steps for all samples[2], which increases the total number of operations but produces a regular computational pattern well suited to GPUs. This modification transforms the main computation, matrix-vector multiplication $G's_i$ for each sample $i = 1, \ldots, N_s$, into a dense matrix-matrix multiplication $G'\left[s_1, \ldots, s_{N_s}\right]$. NVIDIA's cuBLAS[6], linear algebra library highly optimized with Tensor Cores, is employed to perform this computation.

Furthermore, to maximize the utilization of GPU resources, which is underutilized by a single process due to relatively small matrix sizes, Multi-Process Service (MPS)[7] is introduced. MPS allows multiple CUDA kernels to execute concurrently on a single GPU. As a result, inner SMC computations for different θ samples run in parallel without significant code modification, increasing the GPU utilization. MPS also provides dynamic intra-node load balancing. In the outer SMC for θ, synchronization across all processes is required at each iteration. Load imbalance within a GPU is mitigated by MPS allocating larger resources to fewer active processes during the synchronization.

The kernel was integrated into the application and the performance was evaluated on a single node of the Miyabi supercomputer equipped with an NVIDIA GH200 Grace Hopper Superchip[8]. Varying the number of MPS processes per GPU, the performance peaked at 16 MPS processes, achieving 13.40 TFLOPS, which is 20.0% of the peak performance of the FP64 Tensor Cores. A 7.7-fold speedup was achieved compared to the case with one MPS process.

In a weak scaling test, where the number of samples per node was kept constant and the number of nodes was increased from 1 to 128, 92.3% efficiency was achieved from 1 to 128 nodes. The same analysis as the 128 nodes case of Miyabi took 3.5 hours on 1024 nodes of Fugaku with the original CPU implementation. Comparing these results, the GPU implementation on Miyabi achieved a 42.1-fold speedup per node and reduced the energy-to-solution to 18.8%. This speedup is 2.1 times larger than the ratio of peak performance per node between the two supercomputers (3.4TFLOPS on Fugaku and 67.0TFLOPS on Miyabi), highlighting the effectiveness of the methodology designed for GPUs.

3 Concluding Remarks

We developed and demonstrated an efficient, massively parallel Bayesian inference framework for GPU supercomputers. By redesigning the main kernel for GPUs and leveraging MPS, it achieved significant speedup and good scalability. The principles demonstrated here are not limited to this application. They could serve as guide for porting a wide range of similar applications that involve

numerous small, forward simulations, such as ensemble data assimilation and particle-based optimization, to GPU-based computing environments.

References

[1] Jianye Ching and Yi-Chu Chen. 2007. Transitional Markov Chain Monte Carlo Method for Bayesian Model Updating, Model Class Selection, and Model Averaging. *Journal of Engineering Mechanics* 133, 7 (2007), 816–832. doi:10.1061/(ASCE)0733-9399(2007)133:7(816)

[2] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. 2006. Sequential Monte Carlo Samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 68, 3 (05 2006), 411–436. doi:10.1111/j.1467-9868.2006.00553.x

[3] Jin-Chuan Duan and Andras Fulop. 2015. Density-Tempered Marginalized Sequential Monte Carlo Samplers. *Journal of Business & Economic Statistics* 33, 2 (2015), 192–202. doi:10.1080/07350015.2014.940081

[4] Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. 1987. Hybrid Monte Carlo. *Physics Letters B* 195, 2 (1987), 216–222. doi:10.1016/0370-2693(87)91197-X

[5] Kai Nakao, Tsuyoshi Ichimura, Hiroshi Munekane, Tomokazu Kobayashi, Takane Hori, and Kohei Fujita. 2025. Simultaneous Bayesian estimation of multisegment fault geometry and complex slip distribution: application to the 2024 Noto Peninsula earthquake. *Geophysical Journal International* 242, 2 (06 2025), ggaf231. doi:10.1093/gji/ggaf231

[6] NVIDIA. [online]. cuBLAS. https://docs.nvidia.com/cuda/cublas/index.html.

[7] NVIDIA. [online]. Multi-Process Service. https://docs.nvidia.com/deploy/mps/index.html.

[8] NVIDIA. [online]. NVIDIA GH200 Grace Hopper Superchip Architecture. https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper.

[9] RIKEN Center for Computational Science. [online]. Supercomputer Fugaku. https://www.r-ccs.riken.jp/en/fugaku/.

[10] Supercomputing Division, Information Technology Center, The University Tokyo. [online]. Miyabi Supercomputer System. https://www.cc.u-tokyo.ac.jp/en/supercomputer/miyabi/service/.