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

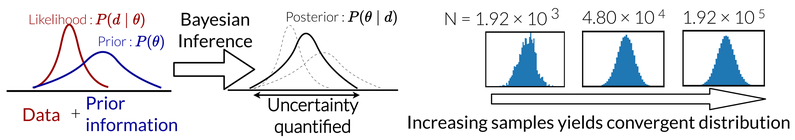
Kai Nakao [1] Tsuyoshi Ichimura [1] Kohei Fujita [1]

[1]Earthquake Research Institute, The University of Tokyo

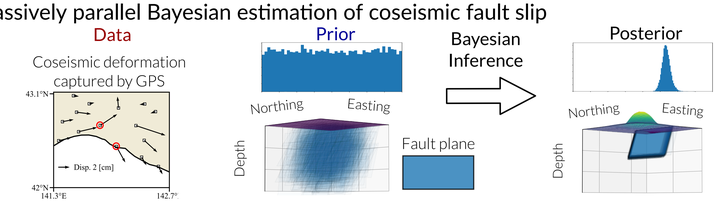
Introduction

Bayesian inference for inverse analysis

- Inverse analysis aims at estimating unobservable model parameters from observable data, and is ubiquitous in science and engineering.
  - e.g., computed tomography, estimation of gravitation-wave source, earthquake fault slip
- **Bayesian inference** is a versatile framework for inverse analysis, integrating data and prior information, and quantifying the uncertainty of the results.
- **A massively parallel computing** is expected to accelerate the analysis.
  - In practice, Monte Carlo sampling is used to obtain the posterior distribution.
  - **Forward simulations for many (e.g., 100,000) samples** lead to huge computational cost.



Massively parallel Bayesian estimation of coseismic fault slip



- The state-of-the-art implementation[1] required 3.5 hours on 1024 nodes of the CPU-based supercomputer Fugaku, unsuitable for GPUs due to many small, imbalanced computations.
- We designed a methodology that **fully exploits high throughput of GPUs**, reducing the time-to-solution to 39.4 minutes on 128 nodes of the GPU-based supercomputer Miyabi.
  - **42.1-fold speedup per node, also energy-to-solution reduced to 18.8%.**

Target Problem

Formulation of Bayesian inference

$$P(\boldsymbol{\theta} \mid \boldsymbol{d}) = \frac{P(\boldsymbol{d} \mid \boldsymbol{\theta})P(\boldsymbol{\theta})}{\int P(\boldsymbol{d} \mid \boldsymbol{s}, \boldsymbol{\theta})P(\boldsymbol{s} \mid \boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}} \quad (1)$$

$$P(\boldsymbol{d} \mid \boldsymbol{\theta}) = \int P(\boldsymbol{d} \mid \boldsymbol{s}, \boldsymbol{\theta})P(\boldsymbol{s} \mid \boldsymbol{\theta})\mathrm{d}\boldsymbol{s} \quad (2)$$

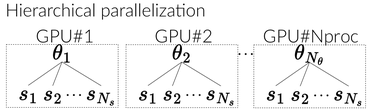
$\boldsymbol{\theta}$: Parameter (Fault geometry)
$\boldsymbol{s}$: Latent variable (Slip distribution)
$\boldsymbol{d}$: Data (Crustal deformation)

Sampling algorithm

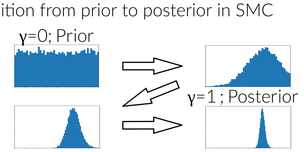
- Sample from the posterior (1) using the likelihood (2) computed by Monte Carlo integration.
- **Sequential Monte Carlo (SMC)** is used for both $\boldsymbol{\theta}$ and $\boldsymbol{s}$, with **hierarchical parallelization**.
  - Outer loop: SMC for $\boldsymbol{\theta}$, Inner loop: SMC for $\boldsymbol{s}$

Algorithm 1: Sequential Monte Carlo
input : Samples from the prior distribution
output: Samples from the posterior distribution
$\gamma \leftarrow 0$;
while $\gamma < 1$ **do**
  Resampling according to likelihood;
  for *each sample* **do**
    Mutation by MCMC (**in parallel**);
  Synchronize processes;
  $\gamma \leftarrow \gamma + \Delta\gamma$;
return;

Hierarchical parallelization



Transition from prior to posterior in SMC



Implementation for GPU

Main component: Hamiltonian Monte Carlo (HMC) for mutation of $\boldsymbol{s}$

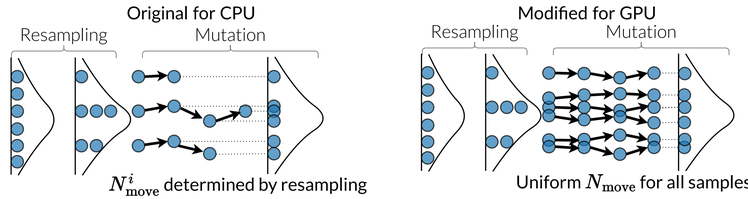
- Only the main component is offloaded to GPUs.
- Repeat the following updates for each sample $\boldsymbol{s}_i\,(i = 1, \ldots, N_{\boldsymbol{s}})$:

$$\begin{cases} \boldsymbol{s}_i & \leftarrow \boldsymbol{s}_i + C\boldsymbol{p}_i\Delta\tau \\ \boldsymbol{p}_i & \leftarrow \boldsymbol{p}_i + (\boldsymbol{d}' - G'\boldsymbol{s}_i)\Delta\tau \end{cases} \quad (3)$$

- C: diagonal matrix, G': dense matrix; **the main computation is the matrix-vector product $G'\boldsymbol{s}_i$.**

Algorithm & Data structure for GPU

- The number of HMC updates was originally sample dependent and caused load imbalance.
- For GPU adaptation, it is **unified for all samples**.
  - Increased total operational count, but suitable for GPU parallelization
- Block variables $\boldsymbol{S} = [\boldsymbol{s}_1 \cdots \boldsymbol{s}_{N_{\boldsymbol{s}}}]$, $\boldsymbol{P} = [\boldsymbol{p}_1 \cdots \boldsymbol{p}_{N_{\boldsymbol{s}}}]$, $\boldsymbol{D}' = [\boldsymbol{d}' \cdots \boldsymbol{d}']$ are introduced and **HMC for all samples is performed concurrently, making the computation dense.**
  - CuBLAS optimized for Tensor Cores is used for the main computation, $G'\boldsymbol{S}$.



Algorithm 2: Original HMC for CPU
for $i = 1, \ldots, N_{\boldsymbol{s}}$ **do**
  for $j = 1, \ldots, N^i_{\mathrm{move}}$ **do**
    for $k = 1, \ldots, N_\tau$ **do**
      $\boldsymbol{s}_i \leftarrow \boldsymbol{s}_i + C\boldsymbol{p}_i\Delta\tau$;
      $\boldsymbol{p}_i \leftarrow \boldsymbol{p}_i + (\boldsymbol{d}' - G'\boldsymbol{s}_i)\Delta\tau$;

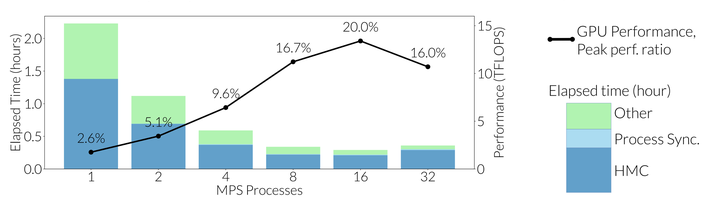
Algorithm 3: Modified HMC for GPU
for $j = 1, \ldots, N_{\mathrm{move}}$ **do**
  for $k = 1, \ldots, N_\tau$ **do**
    $\boldsymbol{S} \leftarrow \boldsymbol{S} + C\boldsymbol{P}\Delta\tau$;
    $\boldsymbol{P} \leftarrow \boldsymbol{P} + (\boldsymbol{D}' - G'\boldsymbol{S})\Delta\tau$;

Utilization of Multi Process Service (MPS)

- Small matrix size of G' and $\boldsymbol{S}$, G': $(200, 200)$, $\boldsymbol{S}$: $(200, 1000)$, underutilizes GPU resources.
- MPS allows multiple CUDA kernels to run concurrently on a single GPU, **enabling concurrent execution of HMC for multiple $\boldsymbol{\theta}$** and **increasing the GPU utilization.**

Performance on Mini Application

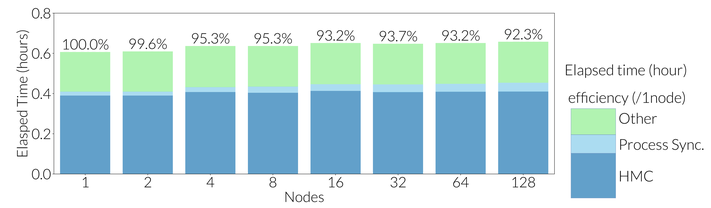
- Bayesian inference with the number of SMC iterations for $\boldsymbol{\theta}$ fixed to 10, on 1 node of Miyabi (NVIDIA GH200 1 module), with various number of MPS processes.
- Number of samples: $N_{\boldsymbol{s}} = 1000, N_{\boldsymbol{\theta}} = 960$. HMC kernel: $N_{\mathrm{move}} = 3, N_\tau = 50$.
- GPU performance is calculated by dividing the GPU operation count by the total elapsed time.
- The maximum GPU performance with **16 MPS processes; 13.40 TFLOPS.**
  - **20.0%** of the peak performance (67 TFLOPS for **FP64 Tensor Cores**)



Performance on Realistic Application

Weak scaling

- Bayesian inference of fault slip with synthetic data was performed on multiple nodes of Miyabi.
- Number of samples:$N_{\boldsymbol{s}} = 1000$, $N_{\boldsymbol{\theta}}$ =(Number of nodes) ×960. MPS processes per GPU: 16.
- **A weak scaling efficiency of 92.3%** was achieved when increasing from 1 to 128 nodes.

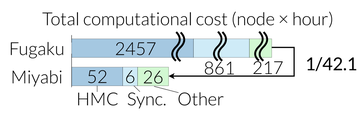


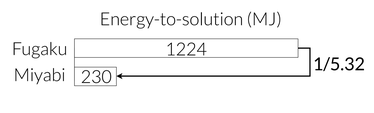
Comparison with original CPU implementation on Fugaku

- The same analysis as the 128 nodes case of Miyabi took 8.2 hours on 1024 nodes of Fugaku with the original implementation.
- **Dynamic intra-node load balancing by MPS** reduced the synchronization cost.
  - Fewer active processes utilize larger GPU resources during synchronization.

Processor		FP64 peak performance (TFLOPS/node)
Miyabi	NVIDIA GH200	67.0
Fugaku	Fujitsu A64FX	3.4

- **This speedup is 2.1 times larger than the ratio of their FP64 peak performances.**





Conclusion

- We developed a **massively parallel Bayesian inference framework for GPU supercomputers**, and performed Bayesian estimation of coseismic fault slip.
  - **Designed a kernel suited for GPU architectures** with uniform and dense computations, and **utilized MPS** to fully exploit GPU resources.
  - Achieved **20.0% of FP64 Tensor Core peak performance on a single GPU** and **92% scalability from 1 to 128 nodes**.
  - Compared to the state-of-the-art CPU implementation, attained **a 42.1-fold speedup per node** and **reduced energy-to-solution to 18.8%.**
- This work is expected to provide **guidance for GPU porting of analyses involving many small forward simulations**, such as various Bayesian inference applications, ensemble data assimilation, and particle-based optimization.

Acknowledgements

This research was conducted using the Supermicro ARS-111GL-DNHR-LCC and FUJITSU Server PRIMERGY CX2550 M7 (Miyabi) at Joint Center for Advanced High Performance Computing (JCAHPC). This work was supported by MEXT, Japan as 'Program for Promoting Researches on the Supercomputer Fugaku' (Large-scale numerical simulation of earthquakes by Fugaku supercomputer, JPMXP1020230213). This work used computational resources of Fugaku provided by the RIKEN Center for Computational Science (Project ID: hp230203, hp240217, hp250230). This work was supported by JSPS KAKENHI, Japan Grant Numbers 23H00213, 25K21686. This work was supported by JST SPRING, Grant Number JPMJSP2108.

References

[1] Kai Nakao, Tsuyoshi Ichimura, Hiroshi Munekane, Tomokazu Kobayashi, Takane Hori, and Kohei Fujita. Simultaneous bayesian estimation of multisegment fault geometry and complex slip distribution: application to the 2024 noto peninsula earthquake. *Geophysical Journal International*, 242(2):ggaf231, 06 2025.