Hierarchical Modelling of Species Communities on LUMI

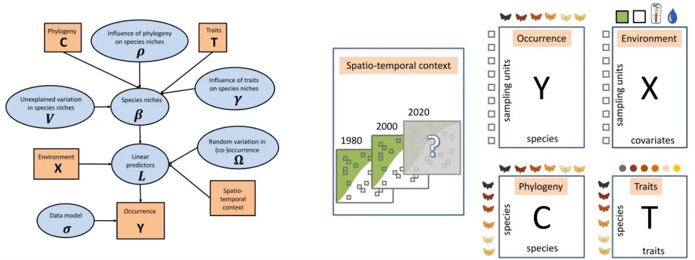
Anis Ur Rahman[1], Gleb Tikhonov[2], Tuomas Rossi[3]

[1] University of Jyväskylä, Finland
[2] University of Helsinki, Finland
[3] CSC, Finland

HMSC for community ecology

Hierarchical Modelling of Species Communities (HMSC) is a model-based approach for ecological community data analysis and forecasting[1].

- **Basic** input data for analyses:
  - a matrix of species occurrences or abundances *Y*,and
  - a matrix of environmental covariates *X*.
- **Optional** input data:
  - species traits *T*,
  - phylogenetic relationships *C*, and
  - spatiotemporal context of the sampling design (*S* and **Π**).



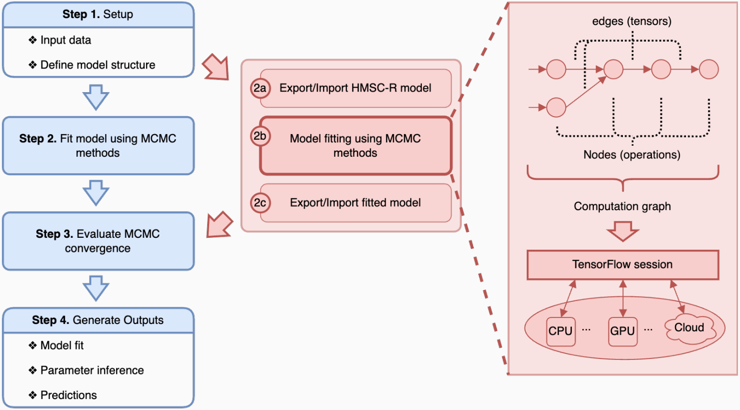
[1]Ovaskainen et al. (2017), "How to make more out of community data? A conceptual framework and its implementation as models and software," Ecology Letters.

Technical overview of HMSC

- Developed by our group in University of Helsinki and University of Jyväskylä with contribution from several external collaborators.
- Under the hood HMSC is a family of statistical models that couple Generalized Linear Model and Latent Factor Model.
- Model fitting is done in Bayesian paradigm with MCMC, namely block-Gibbs sampler.
- Computational load is primarily through linear algebra operations.
- Hmsc-R package is published in 2020 and has been positively accepted by community ecologists.
- The computational core of Hmsc-R is implemented in R with CPU execution on local machine in mind. Some linear algebra routines may benefit greatly from dedicated co-processor (as in M1/M2).
- Scalability for analysis of larger datasets has been identified as a major an issue, as the model fitting times can become unreasonably large.

HMSC has moved to GPU

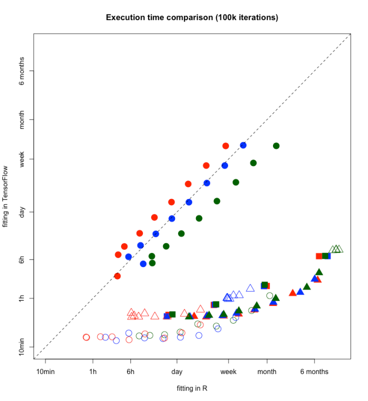
- In 2022 we started developing GPU-compatible implementation Hmsc-HPC
- TensorFlow was selected as the best trade-off choice for backend
- Designed as replacement of computation core in Hmsc-R
- Main re-coding is at the end, debugging/testing is ongoing
- x1000 times speed-up for large models with NVIDIA Volta V100



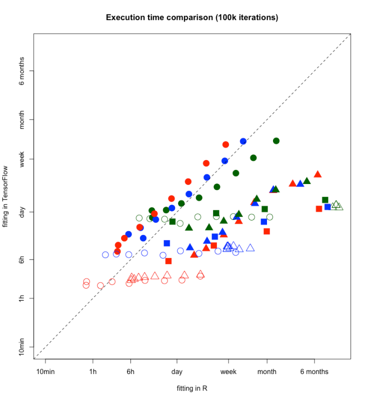
Powering Up with LUMI Supercomputer

- Why LUMI?
  - **Unprecedented Computational Power:** LUMI offers unparalleled computing capabilities, aligning with our vision for cutting-edge ecological research.
  - **Optimization Opportunities:** The move to LUMI provides a chance to optimize our workflow for enhanced analysis.
  - **Large VRAM GPUs:** The model fitting algorithm repeatedly operates with potentially very large objects and lacks remedy like mini-batching.
- Anticipated Benefits:
  - **Enhanced Scalability:** Leveraging LUMI's capabilities to scale our analyses for handling extensive ecological datasets.
  - **Efficiency Boost:** Expecting substantially reduced computation times, fueling research productivity.

Puhti vs LUMI — initial performance comparison



Hmsc-R vs Hmsc-HPC in Puhti



Hmsc-R vs Hmsc-HPC in LUMI

Currently NVIDIA V100 clearly outperforms the more powerful AMD MI250x.

- Why? Are some specific operations particularly slow?
- How this can be circumvented?

Advancing HMSC-HPC with AMD Profiling and Multi-GPU

1. Profiling on AMD Graphics Cards:
    - **Diverse Hardware Assessment:** Evaluating the potential of newer AMD GPUs to further optimize ecological data analysis.
    - **Performance Insights:** Profiling and benchmarking on AMD GPUs to understand their unique capabilities and characteristics.
2. Multi-GPU Development:
    - **Our Goal:** Advancing ecological data analysis through multi-GPU parallelism.
    - **Enhanced Scalability:** Developing a version of HMSC-R that efficiently utilizes multiple GPUs for accelerated model fitting.

Expected Outcomes

- Anticipated Outcomes:
  - **Improved Efficiency:** Expecting to significantly enhance the efficiency of ecological data analysis.
  - **Faster Execution:** Aiming for considerably faster execution times for complex models.
  - **Resource Optimization:** Identifying ways to better utilize available computational resources.
- Contributions:
  - **Advancing Ecological Research:** Our work contributes to the advancement of community ecology research by enabling the analysis of larger and more complex datasets.
  - **High-Performance Computing:** Demonstrating the potential of high-performance computing and GPU acceleration in ecological modeling.
  - **Knowledge Sharing:** Sharing our findings and methodologies with the community to benefit other researchers.

Preliminary Project Timeline

- Day 1:
  1. Transition to LUMI supercomputer and initial setup.
  2. Learning and exploring AMD GPUs.
  3. Preliminary bottleneck identification.
  4. Brainstorming prominent solutions.
- Day 2:
  1. Performance optimization efforts.
  2. Performance debugging and profiling.
  3. Thoughtful assessment of profiling results.
  4. Finalizing optimizations and conducting performance testing.
- Day 3:
  1. Trying the multi-GPU version.
  2. Formulating the vision for post-hackathon development.
  3. Preparing presentation and documentation for the hackathon presentation.

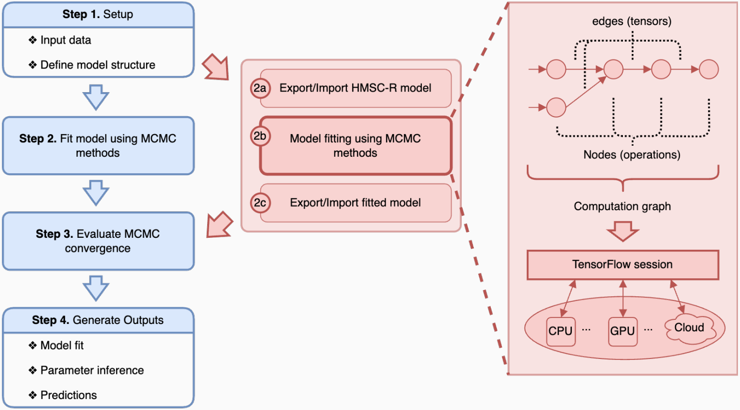
Hierarchical Modelling of Species Communities on LUMI

Anis Ur Rahman[1], Gleb Tikhonov[2], Tuomas Rossi[3]

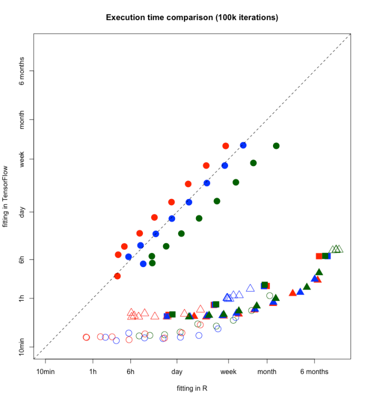
[1] University of Jyväskylä, Finland
[2] University of Helsinki, Finland
[3] CSC, Finland

HMSC has moved to GPU

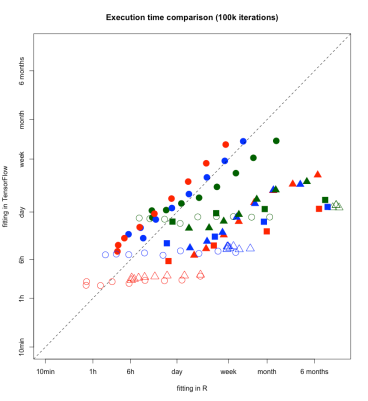
- In 2022 we started developing GPU-compatible implementation Hmsc-HPC
- TensorFlow was selected as the best trade-off choice for backend
- Designed as replacement of computation core in Hmsc-R
- Main re-coding is at the end, debugging/testing is ongoing
- x1000 times speed-up for large models with NVIDIA Volta V100



R vs Puhti vs LUMI — initial performance comparison



Hmsc-R vs Hmsc-HPC in Puhti

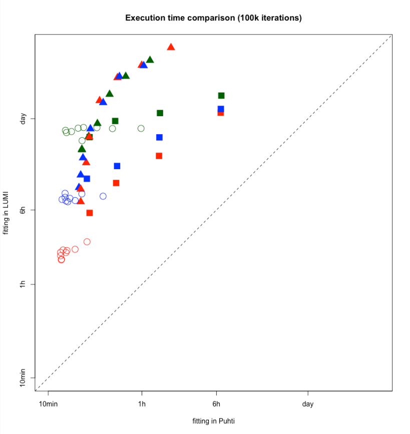


Hmsc-R vs Hmsc-HPC in LUMI

Currently NVIDIA V100 clearly outperforms the more powerful AMD MI250x.

- Why? Are some specific operations particularly slow?
- How this can be circumvented?

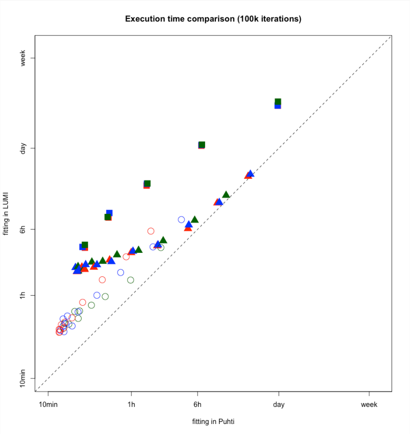
Puhti vs LUMI — initial performance comparison



Puhti (x) vs LUMI (y)

- Puhti - TF 2.12, LUMI - TF 2.11

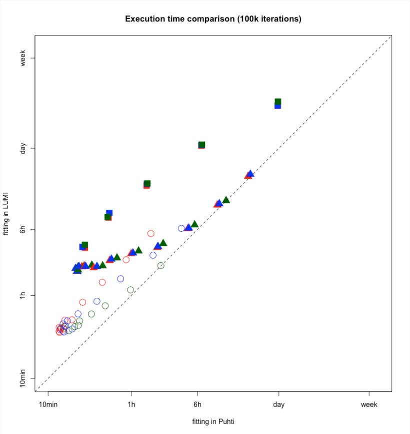
Puhti vs LUMI — matching TF versions



Puhti (x) vs LUMI (y)

- Some operations are slow
- Cholesky factorization, Einstein sum, ”scatter nd”

Puhti vs LUMI — optimizing code

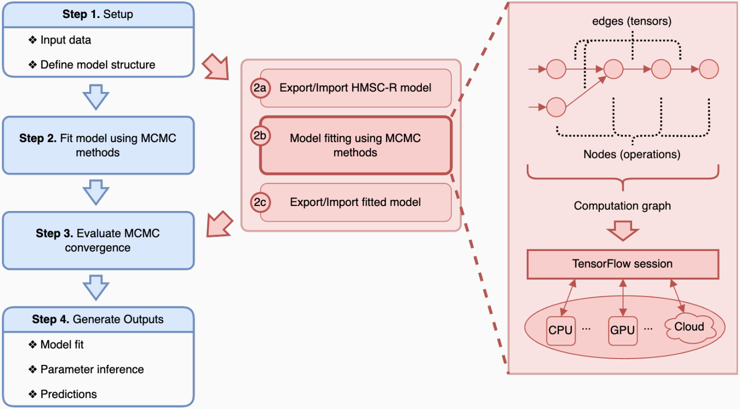


Puhti (x) vs LUMI (y)

- Cholesky factorization is hard to avoid
- Einstein sum $\longrightarrow$ is contradictory

Multiple GPUs

- Algorithm supports parallelization to a few independent threads
- Can be spread to different batch jobs, but i/o, preprocessing is common
- Alternatively idea to utilize multiple GPUs in a single node
- Code for threading is ready, but bugged, no speed-up observed — why?



Conclusions and next steps

LUMI performance is **sometimes** equally good as Puhti. Are we happy about this — probably not.

- Get in touch about problematic operations
- Finalize multi-job, multi-thread implementation
- More thoughtful revision of AMD vs NVIDIA performance
- Test equivalent code writing variants

Work Plan

Total weeks: 26 (or 6 months)

- **Task 1:** Improve Parallelism in Existing Implementation
  - Analyze the existing implementation and identify bottlenecks (2 weeks)
  - Optimize parallelization techniques and GPU resource utilization (2 weeks)
- **Task 2:** Intranode Implementation, Evaluation, and Improvement
- **Task 3:** Internode Implementation, Evaluation, and Improvement
- **Task 4:** Introduce LUMI-G and AMD-Specific Optimizations
- **Task 5:** Implement GPU-optimized Operations Missing in TensorFlow Library
- **Optional task:** Testing, Evaluation, and Documentation

Work Plan

Total weeks: 26 (or 6 months)

- **Task 1:** Improve Parallelism in Existing Implementation
- **Task 2:** Intranode Implementation, Evaluation, and Improvement
    - Implement intranode parallelization and optimize data transfers (3 weeks)
    - Evaluate performance and identify areas for improvement (2 weeks)
    - Optimize and refine intranode implementation (1 weeks)
- **Task 3:** Internode Implementation, Evaluation, and Improvement
- **Task 4:** Introduce LUMI-G and AMD-Specific Optimizations
- **Task 5:** Implement GPU-optimized Operations Missing in TensorFlow Library
- **Optional task:** Testing, Evaluation, and Documentation

Work Plan

Total weeks: 26 (or 6 months)

- **Task 1:** Improve Parallelism in Existing Implementation
- **Task 2:** Intranode Implementation, Evaluation, and Improvement
- **Task 3:** Internode Implementation, Evaluation, and Improvement
  - Design and implement internode parallelization and communication protocols supported for TensorFlow (3 weeks)
  - Evaluate scalability and performance in a distributed environment (2 weeks)
  - Identify and address any scalability issues (1 weeks)
- **Task 4:** Introduce LUMI-G and AMD-Specific Optimizations
- **Task 5:** Implement GPU-optimized Operations Missing in TensorFlow Library
- **Optional task:** Testing, Evaluation, and Documentation

Work Plan

Total weeks: 26 (or 6 months)

- **Task 1:** Improve Parallelism in Existing Implementation
- **Task 2:** Intranode Implementation, Evaluation, and Improvement
- **Task 3:** Internode Implementation, Evaluation, and Improvement
- **Task 4:** Introduce LUMI-G and AMD-Specific Optimizations
  - Study LUMI-G architecture and AMD GPU features (1 weeks)
  - Integrate AMD ROCm library and CuPy for optimizations (2 weeks)
  - Apply other AMD-specific optimizations to the implementation (1 weeks)
- **Task 5:** Implement GPU-optimized Operations Missing in TensorFlow Library
- **Optional task:** Testing, Evaluation, and Documentation

Work Plan

Total weeks: 26 (or 6 months)

- **Task 1:** Improve Parallelism in Existing Implementation
- **Task 2:** Intranode Implementation, Evaluation, and Improvement
- **Task 3:** Internode Implementation, Evaluation, and Improvement
- **Task 4:** Introduce LUMI-G and AMD-Specific Optimizations
- **Task 5:** Implement GPU-optimized Operations Missing in TensorFlow Library
  - Identify critical missing GPU-optimized operations (1 weeks)
  - Develop CUDA kernels for Cholesky decomposition, sparse ops, etc. (4 weeks)
  - Integrate GPU-optimized operations into the implementation (1 weeks)
- **Optional task:** Testing, Evaluation, and Documentation

Work Plan

Total weeks: 26 (or 6 months)

- **Task 1:** Improve Parallelism in Existing Implementation
- **Task 2:** Intranode Implementation, Evaluation, and Improvement
- **Task 3:** Internode Implementation, Evaluation, and Improvement
- **Task 4:** Introduce LUMI-G and AMD-Specific Optimizations
- **Task 5:** Implement GPU-optimized Operations Missing in TensorFlow Library
- **Optional task:** Testing, Evaluation, and Documentation
    - Perform extensive testing and benchmarking
    - Evaluate performance, accuracy, and scalability
    - Document the ported implementation, optimizations, and findings