Unlocking GPU Potential: HMSC and LUMI-G for Ecology

Anis Ur Rahman[1], Gleb Tikhonov[2], Otso Ovaskainen[1], Jari Oksanen[2]

[1] University of Jyväskylä, Finland
[2] University of Helsinki, Finland
email:anis.u.rahman@jyu.fi

Table of contents

Introduction and motivation

The context

The **HORIZON-INFRA-2021-TECH-01** project **"Biodiversity Digital Twin for Advanced Modelling, Simulation and Prediction Capabilities."**

European research project:

- focused on **development of pipelines** for processing of massive biodiversity data with **LUMI**, Europe's fastest petascale supercomputer.

Contributions:

1. to better **observe** spatio-temporal changes in biodiversity,
2. to push **limits of** predictive biodiversity modelling, and
3. to **provide** infrastructure to drive long-term biodiversity research.

The context

The **HORIZON-INFRA-2021-TECH-01** project **"Biodiversity Digital Twin for Advanced Modelling, Simulation and Prediction Capabilities."**

European research project:

- focused on **development of pipelines** for processing of massive biodiversity data with **LUMI**, Europe's fastest petascale supercomputer.

Contributions:

1. to better **observe** spatio-temporal changes in biodiversity,
2. to push **limits of** predictive biodiversity modelling, and
3. to **provide** infrastructure to drive long-term biodiversity research.

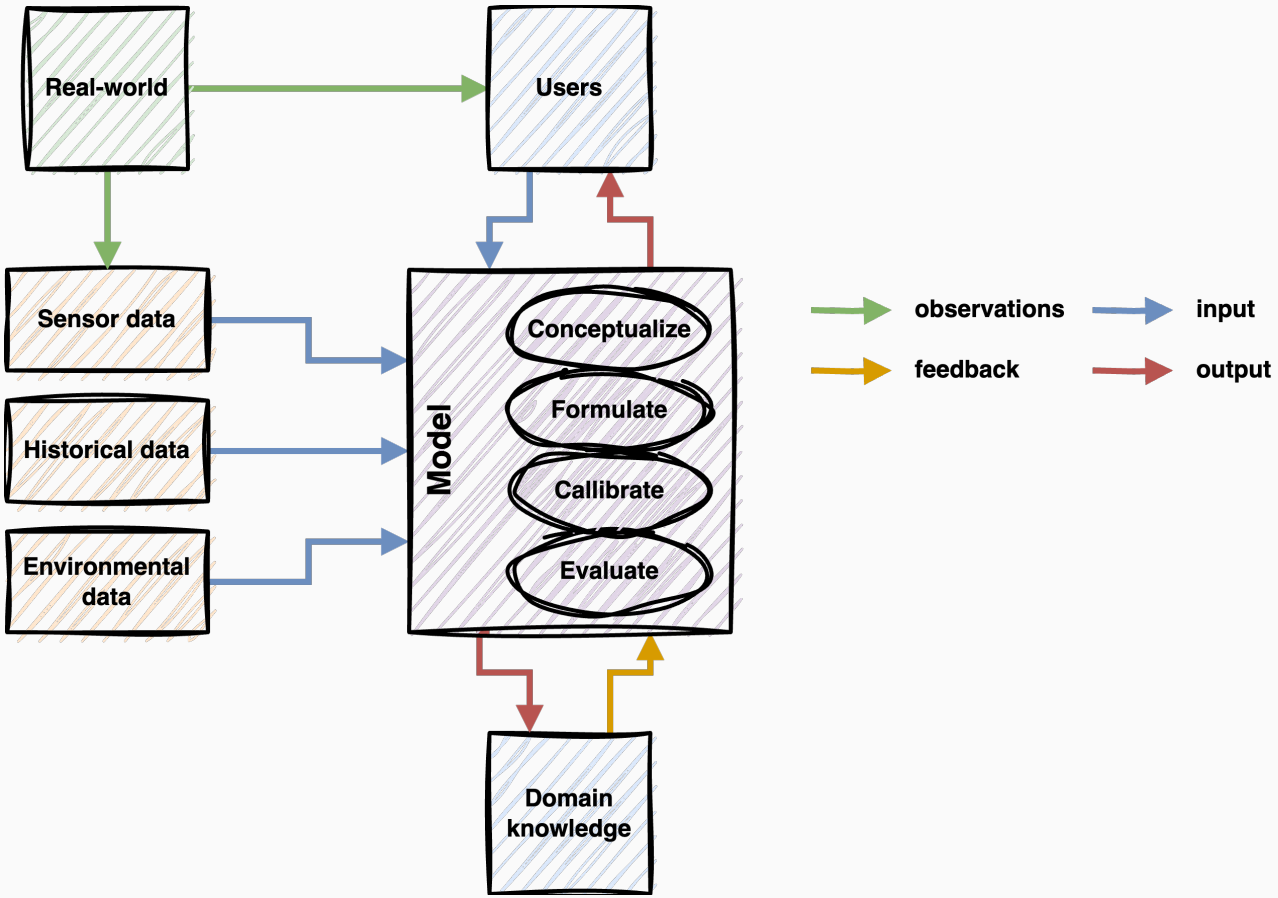
LIFEPLAN – A Planetary Inventory of Life:

- **establish** current state of biodiversity across the globe, and
- **generate** accurate predictions of its future state under future scenarios.

Approaches to "efficient" predictive modelling

A real need:

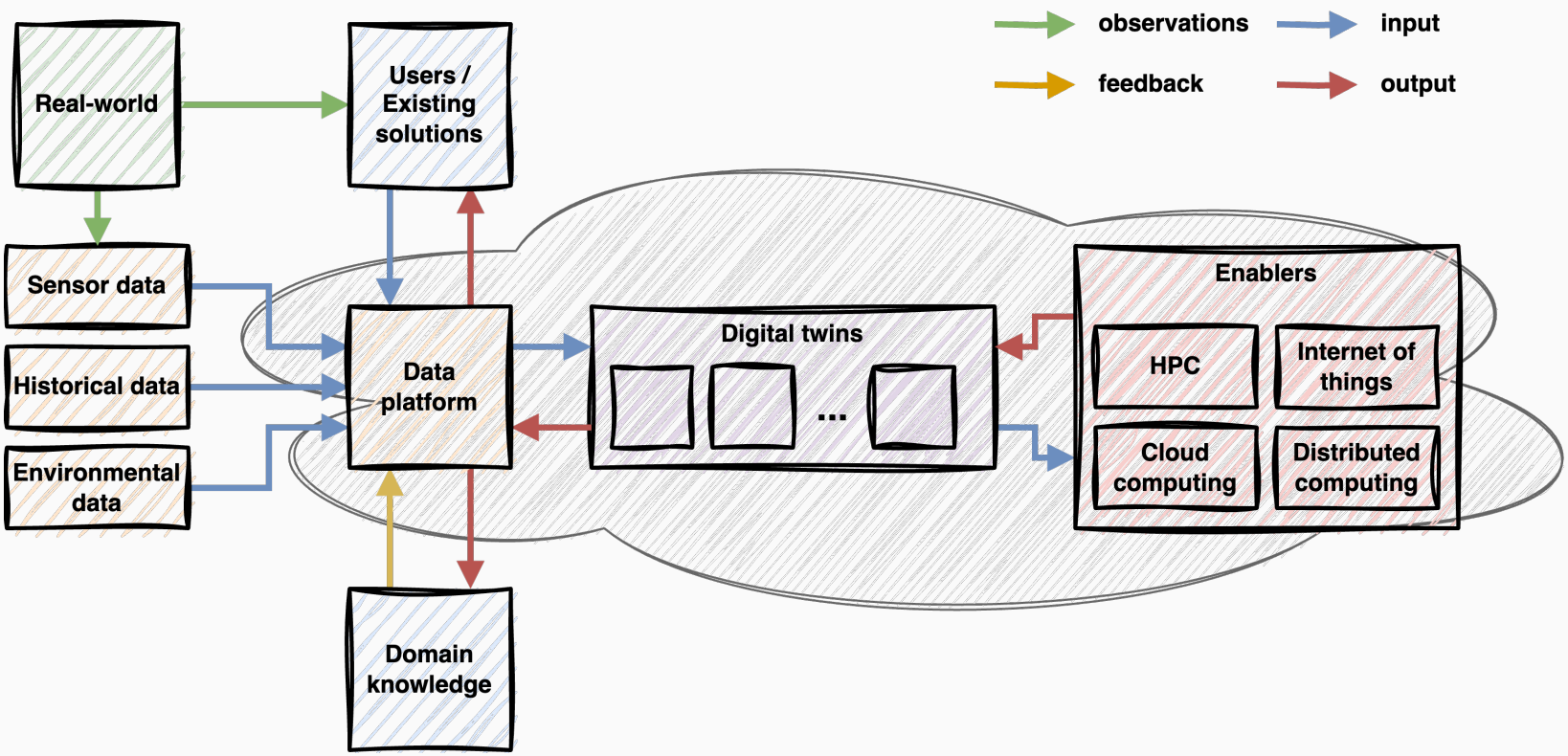
- **Optimized** operations/components for predictive modelling
- More traditional data processing for **large-scale** biodiversity data
- **Digital twins** for ecological research



Approaches to "efficient" predictive modelling

A real need:

- **Optimized** operations/components for predictive modelling
- More traditional data processing for **large-scale** biodiversity data
- **Digital twins** for ecological research



Approaches to "efficient" predictive modelling

A real need:

- **Optimized** operations/components for predictive modelling
- More traditional data processing for **large-scale** biodiversity data
- **Digital twins** for ecological research

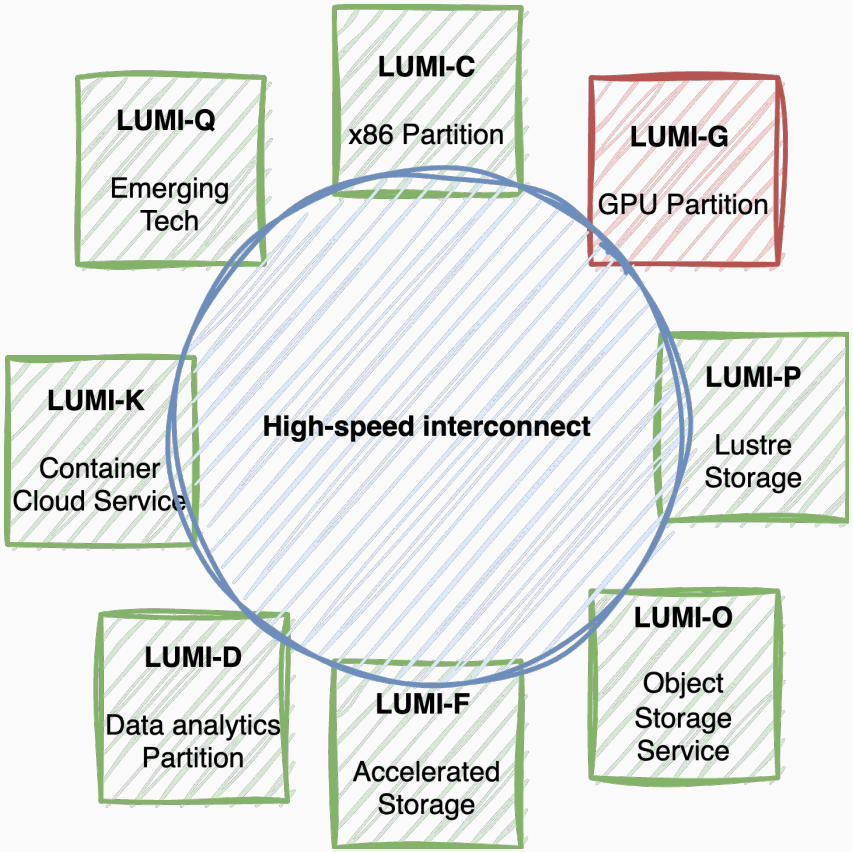
Research questions

1. Can the computational **performance** of species distribution modeling be significantly improved using GPUs?
2. How does parallelizing it impact the **accuracy** and **precision**?
3. What are the potential **challenges** and **bottlenecks** of the parallelization, and how can they be mitigated?

The hardware platform – LUMI

Fastest in Europe and **3rd fastest** globally (Top500 list published in Nov 2022).

- **550 petaflops** theoretical peak performance
- 32 TB memory
- 7 PB flash storage with extreme 2 TB/s bandwidth
- 30 PB object storage for staging area
- 80 PB parallel file system
- Interconnection network:
  - four 200 Gbit/s network interconnect cards

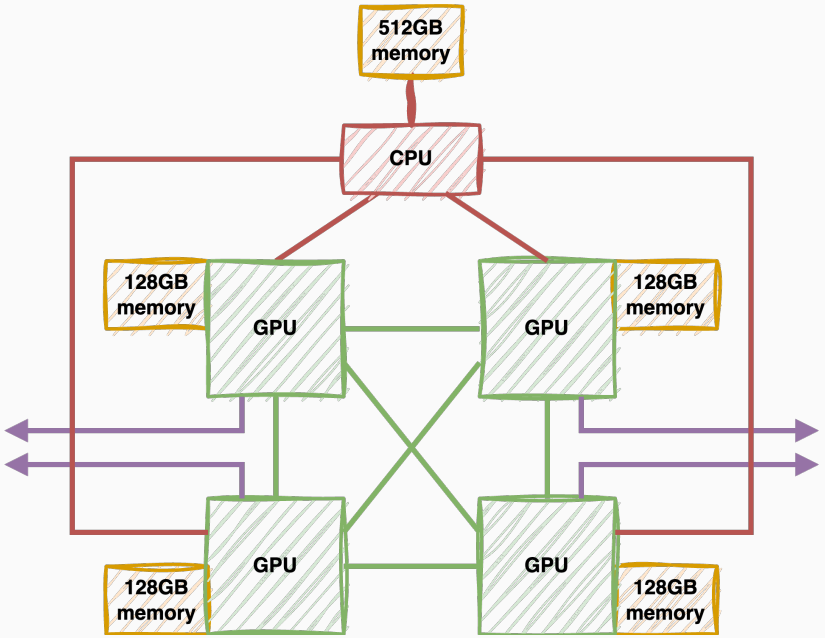


The hardware platform – LUMI

Fastest in Europe and **3rd fastest** globally (Top500 list published in Nov 2022).

LUMI-G: GPU partition

- 2560 nodes, each node includes:
  - one 64 core AMD Trento CPU and four **AMD MI250X GPUs**
- Concept LUMI-G node:
  - **GPU-first** system with coherent unified memory, and
  - compute accelerator, not a rendering GPU.



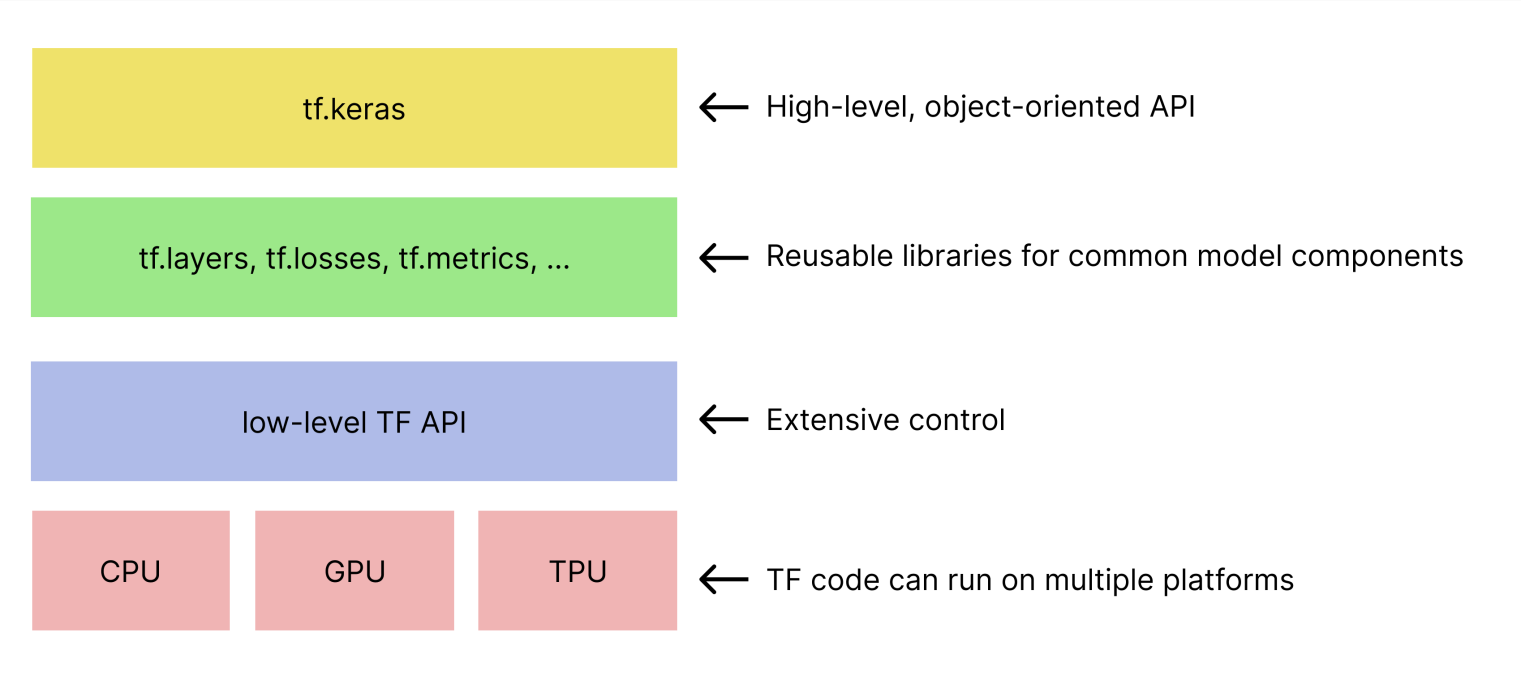
Why is then GPU computing not more widespread?

1. **Diminishing returns**
   - existing CPU-optimized codebase
   - expanding existing code makes more sense
   - porting code is complicated
2. **Lack of expertise**
   - GPU programming paradigm is different
   - New programming languages (CUDA, OpenCL) and frameworks (TensorFlow, PyTorch)
   - more effort required to achieve previous performance
3. **Lack of tools**
   - extensive set of tools CPU
   - more support for distributed computing

What is TensorFlow

TensorFlow is an **end-to-end** open source framework for machine learning.

- capable of running on CPUs and/or GPUs



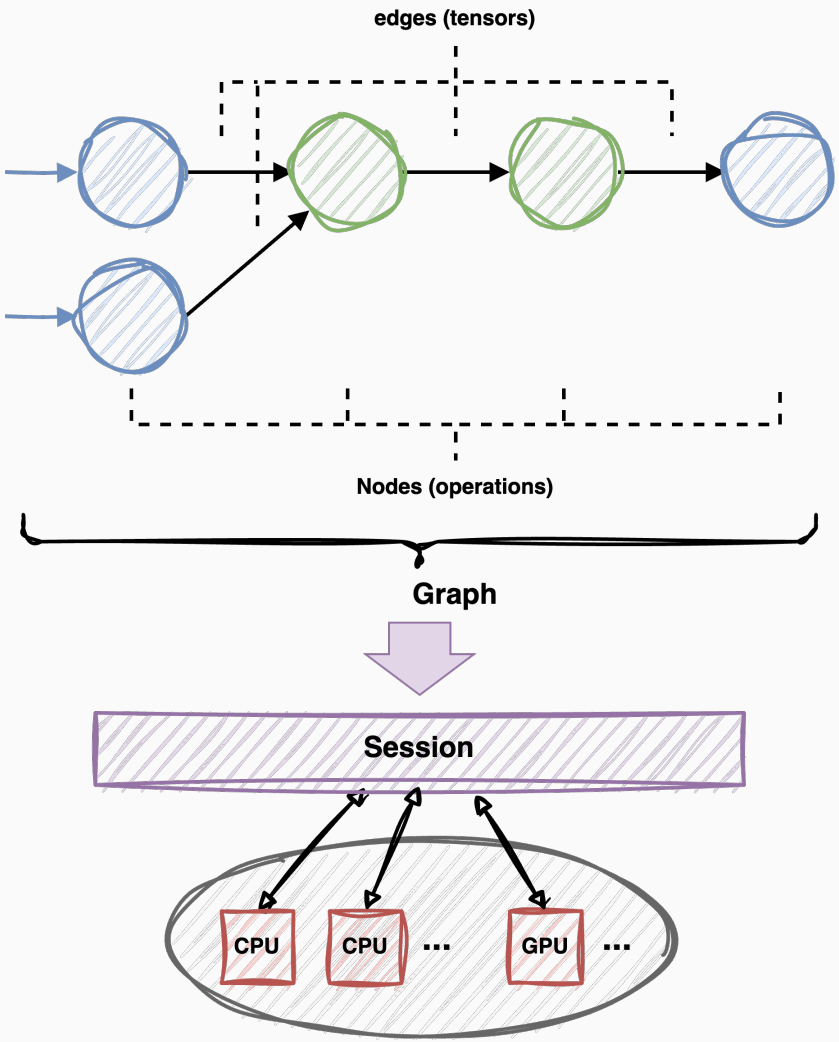
What is TensorFlow

TensorFlow is an **end-to-end** open source framework for machine learning.

- capable of running on CPUs and/or GPUs

Design principles:

1. **Dataflow graphs** of primitive operators
2. **Deferred execution:** Symbolic dataflow graph
3. **Heterogeneous** accelerators

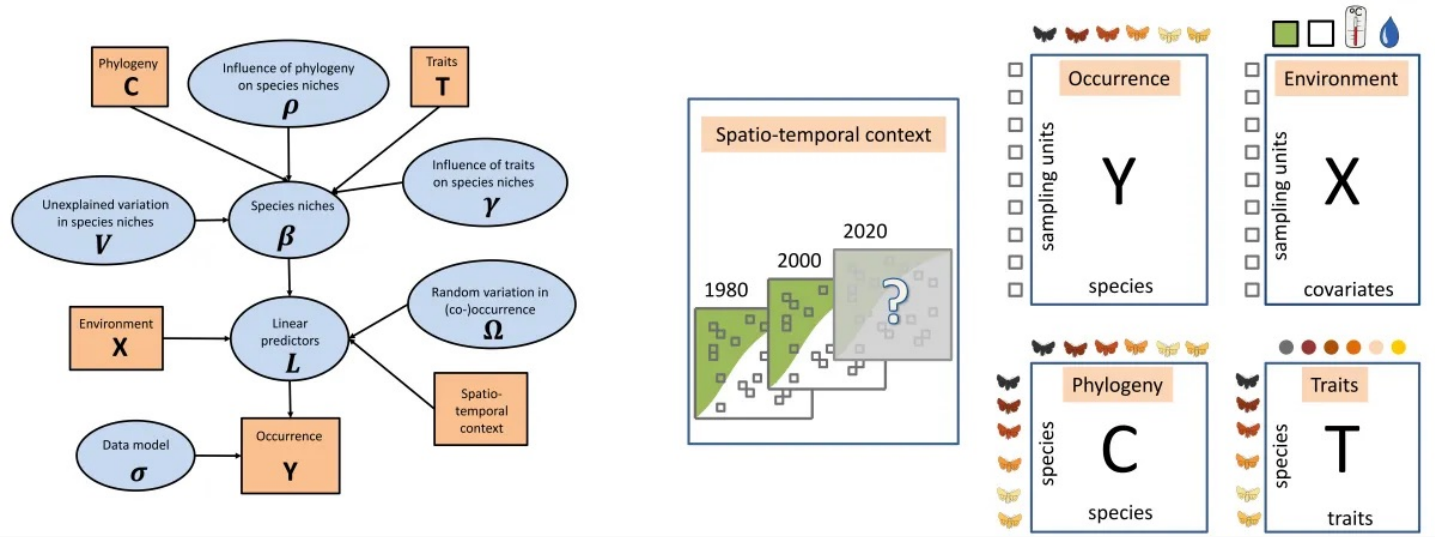


The task

HMSC for community ecology

Hierarchical Modelling of Species Communities (HMSC) is a model-based approach for analyzing community ecological data.

- **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).

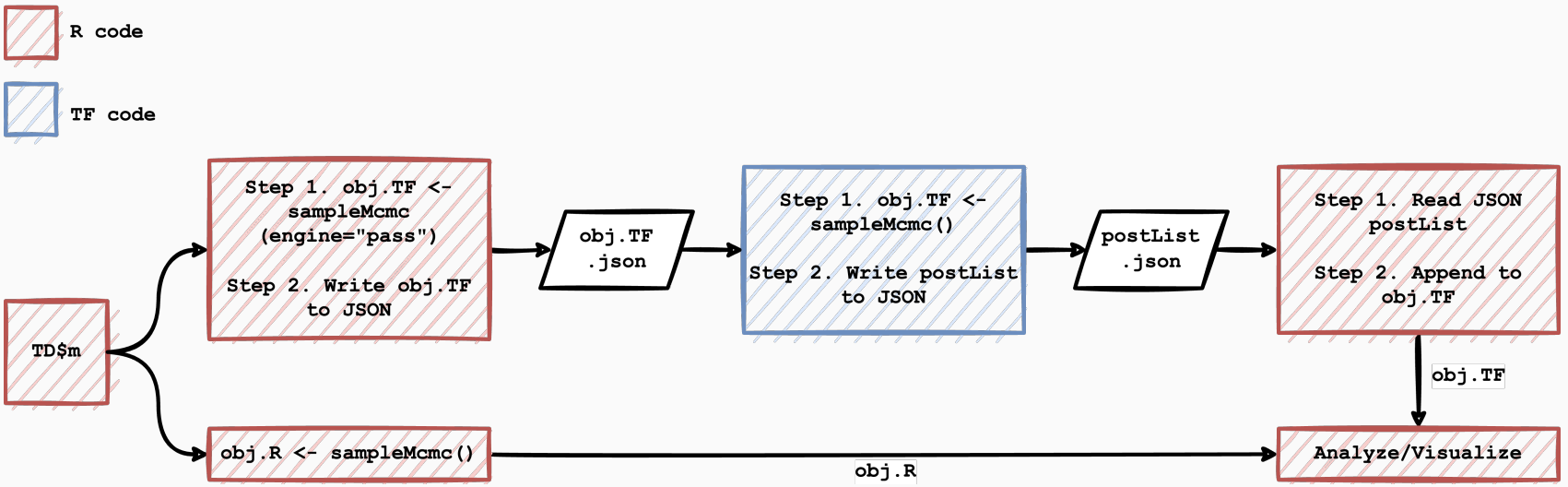


Ovaskainen et al. (2017), "How to make more out of community data? A conceptual framework and its implementation as models and software," Ecology Letters.

The environment

The task:

1. **Implement** parameter updaters using Tensorflow,
2. **Generate** posterior samples based on Gibbs sampler, and
3. **Evaluate** a sample models.



The environment

The task:

1. **Implement** parameter updaters using Tensorflow,
2. **Generate** posterior samples based on Gibbs sampler, and
3. **Evaluate** a sample models.

Figures of interest:

1. **Performance.** Assess the performance improvement achieved by parallelizing the R-package for species distribution modeling on GPUs using TensorFlow.
2. **Accuracy.** Evaluate the impact of parallelization on the accuracy of species distribution modeling predictions.
3. **Scalability.** Assess the scalability of the parallelized R-package on GPUs using TensorFlow.
4. **Comparative.** Conduct a comparative evaluation of the parallelized R-package with other existing species distribution modeling packages or frameworks.

Methodology

We evaluate parallel HMSC implementation for:

model	sampling units	species	covariates	traits	random levels	pylogeny	spatial method
M1	48	160	3	3	5	[x]	Null
M2	1974	284	6	1	2	[x]	Null
M3	5097	68	5	1	3	[ ]	Null
M4	2836	2	8	1	1	[x]	NNGP
M5	2836	2	8	1	1	[x]	GPP
M6	2326	81	4	1	2	[ ]	Full
M7	2326	81	39	1	2	[ ]	Full

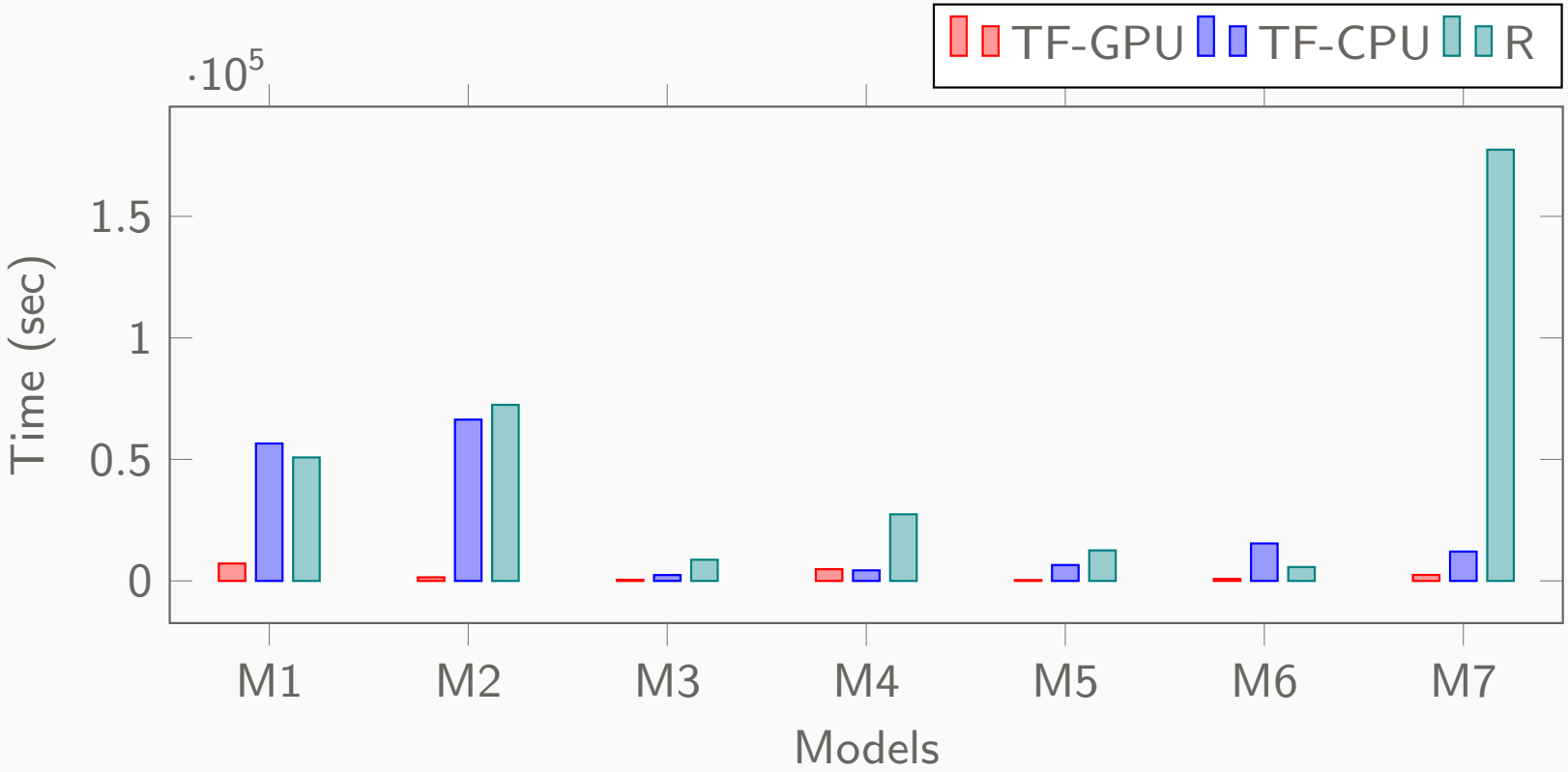
Experimental runs: on **Puhti AI partition**

- 80 GPU nodes with total peak performance of 2.7 petaflops.
- Each node has four **Nvidia Volta V100 GPUs** with 32 GB of memory each.

Configurations: nChain = 8; nSamples = 250; thinning = 10; transient = 2500 (total 5000 iterations per chain)

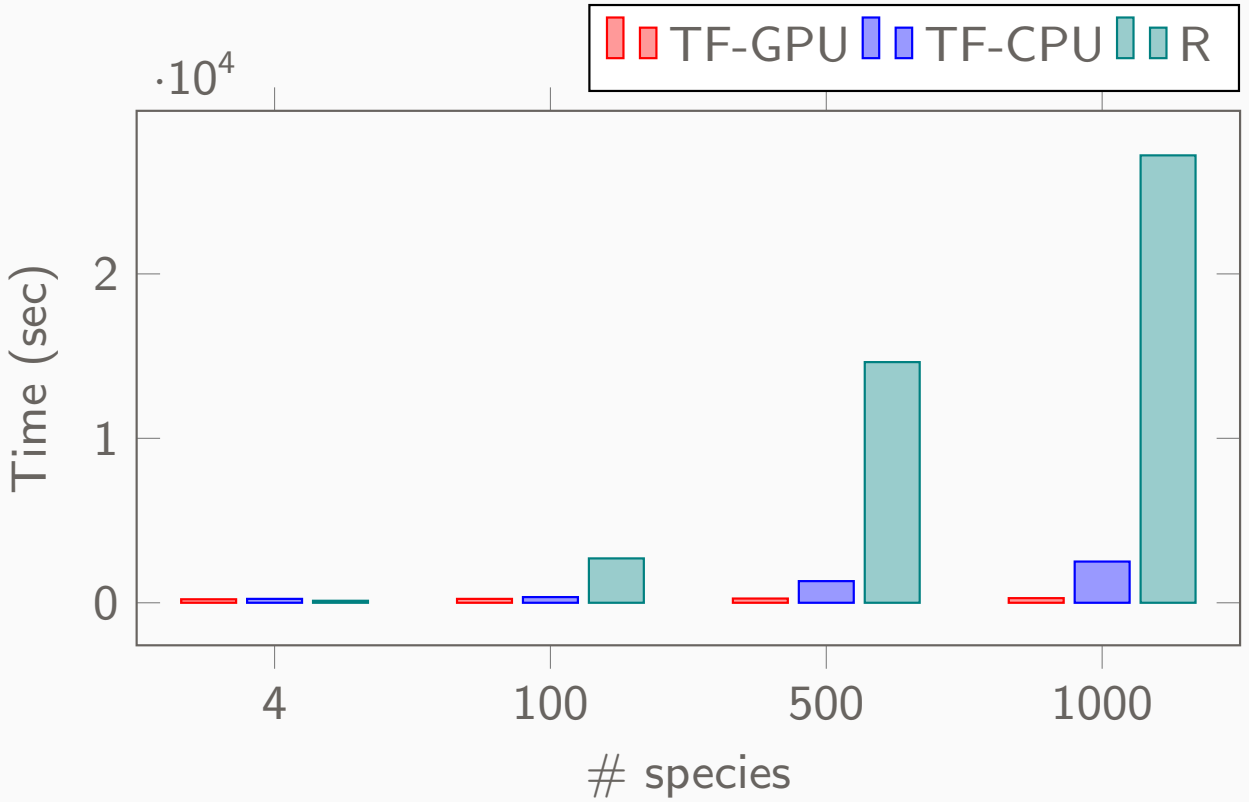
Results

Performance comparison: CPU-GPU (on Puhti)



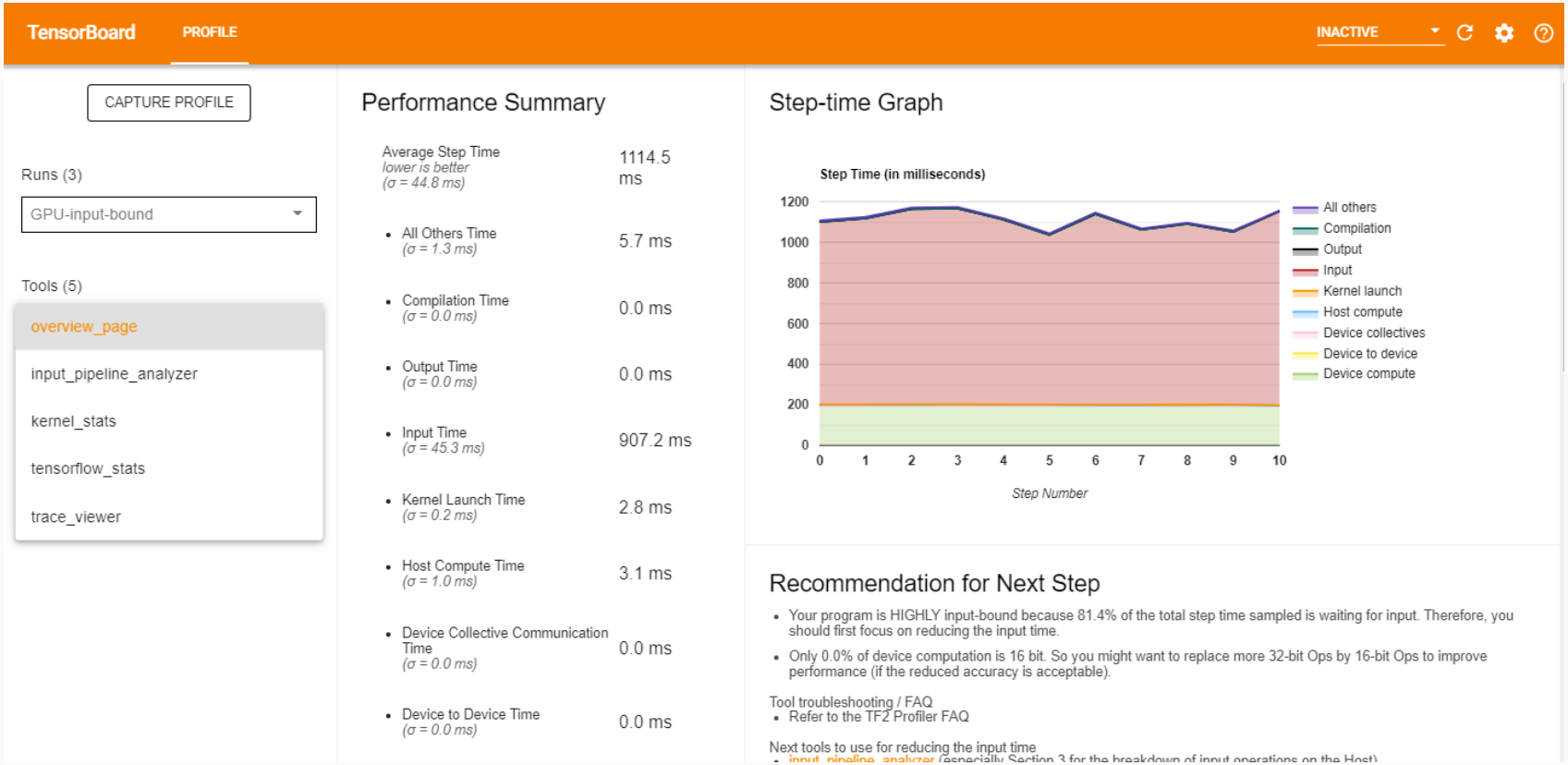
model	TF-GPU (s)	TF-CPU (s)	R (s)	speedup
M1	7159.1	56542.5	50806.9	7x
M2	1481.6	66367.5	72435.4	49x
M3	483.3	2415.3	8712.1	18x
M4	4828.1	4352.5	27403.4	6x
M5	357.2	6539.6	12562.7	35x
M6	863.0	15409.3	5708.8	7x
M7	2474.3	12078.5	177423.2	72x

Scalability - Varying the number of species



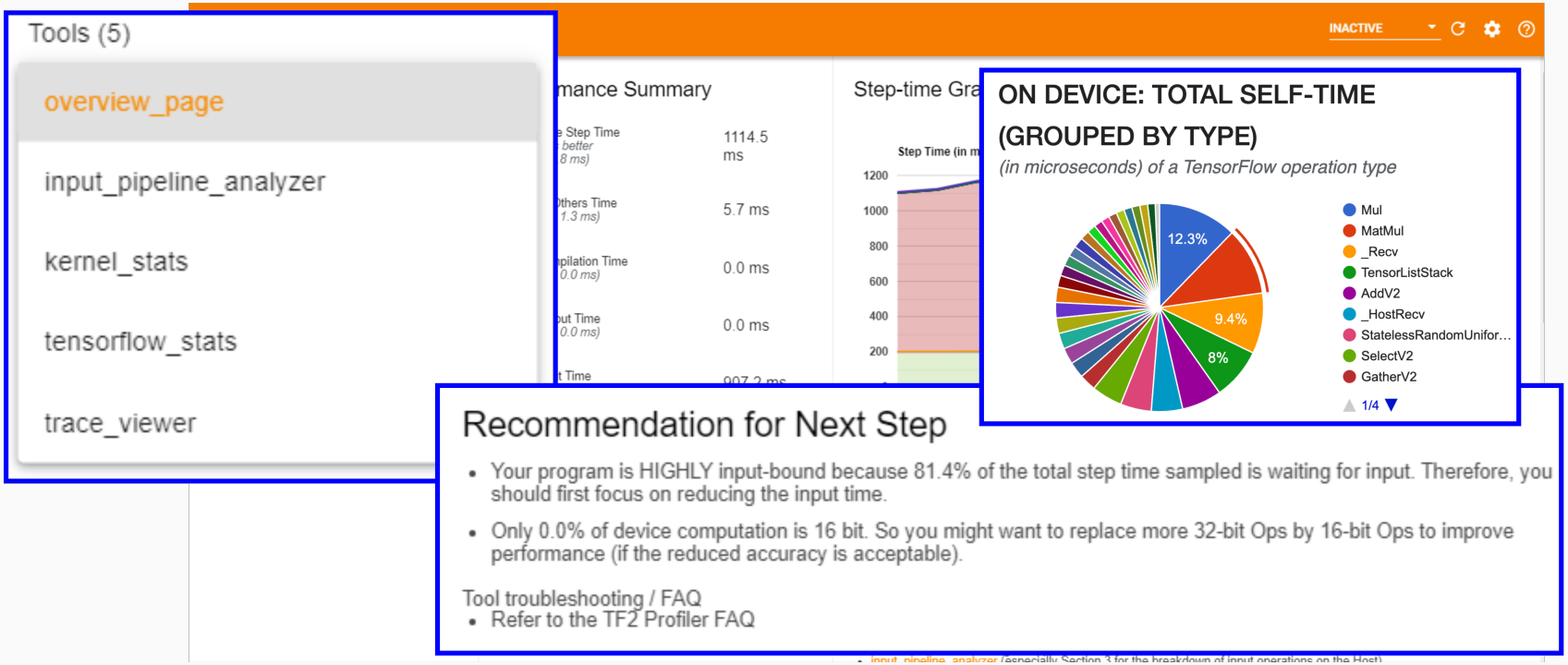
# species	TF-GPU (s)	TF-CPU (s)	R (s)	speedup
4	212.9	237.1	122.4	-
100	236.8	345.5	2698.5	11x
500	257.4	1316.6	14635.2	57x
1000	281.2	2506.5	27208.0	97x

Profiling: Analyzing Profiles in TensorBoard



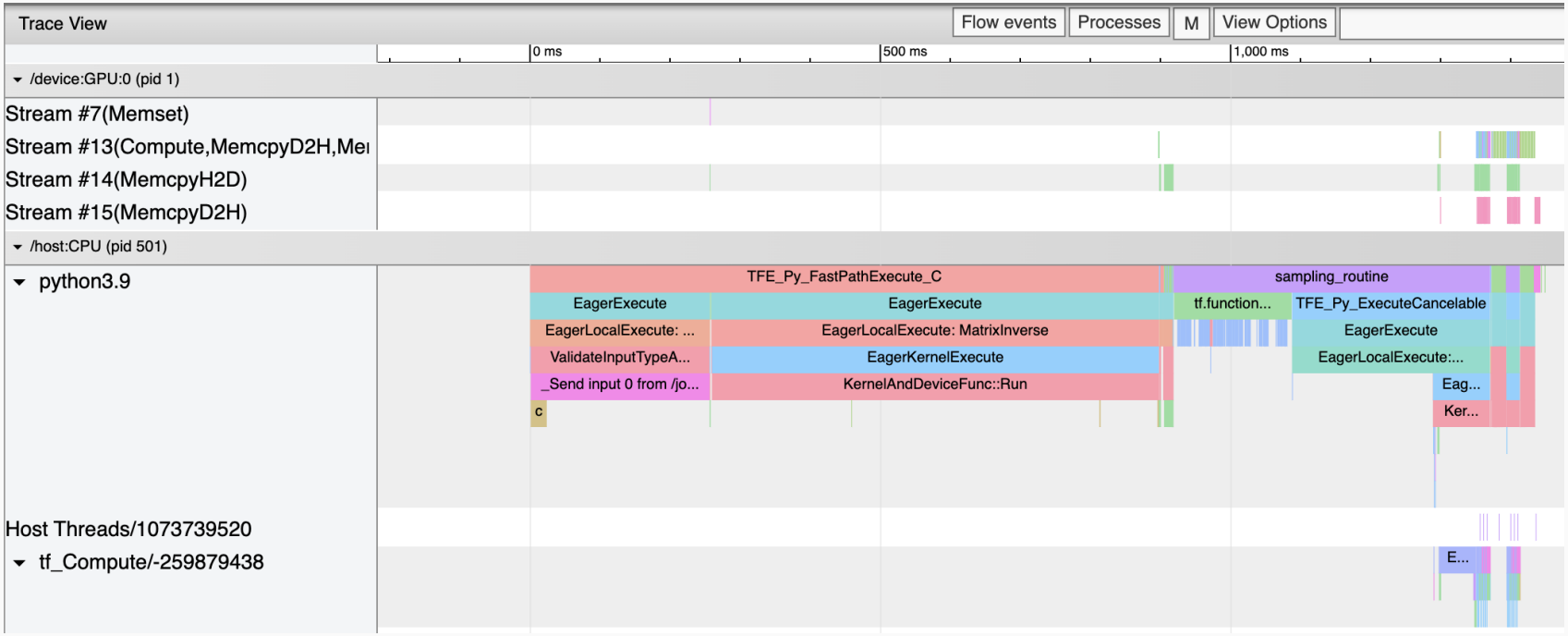
Performance improvement heuristics are often provided with links to more detailed information.

Profiling: Analyzing Profiles in TensorBoard



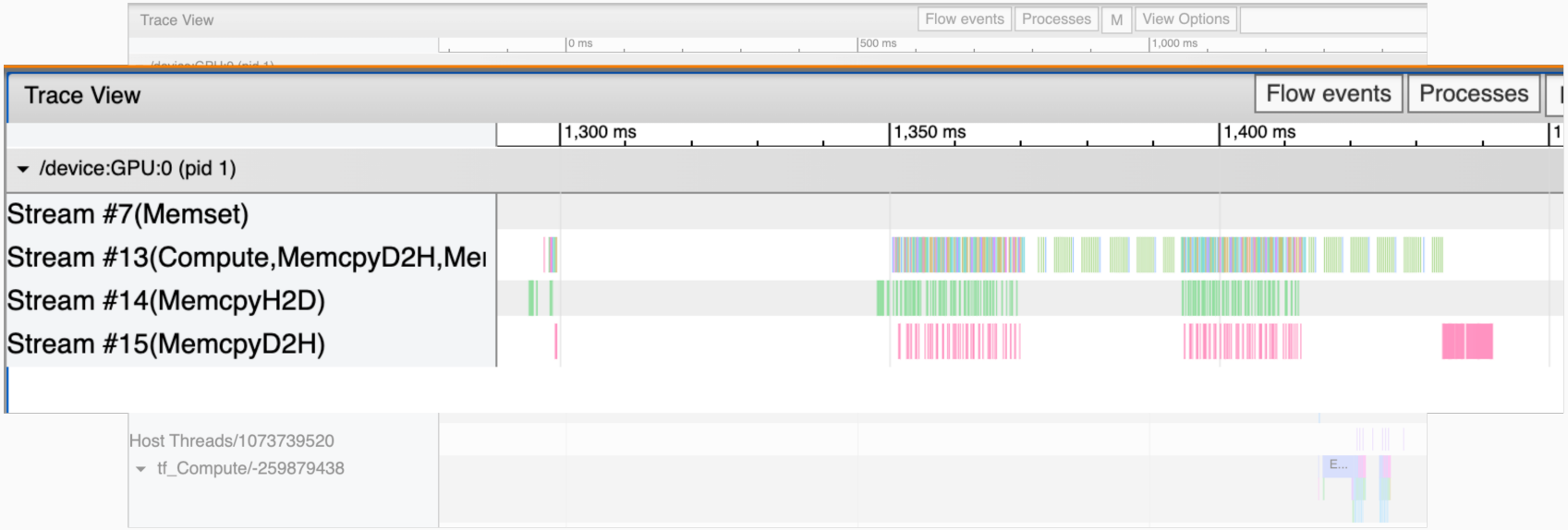
Performance improvement heuristics are often provided with links to more detailed information.

Profiling: Tracing in TensorBoard



Use the trace_viewer to get an overall timeline of TF program.

Profiling: Tracing in TensorBoard



Use the trace_viewer to get an overall timeline of TF program.

Profiling: Common Performance Considerations

1. I/O
    - Use designated TF functions
    - Multithreading, for improved internode performance (future plan)
2. CPU to/from GPU data copies
    - Rewrite code with TF tensors
    - Overlap copy and computation
3. Precision
    - Consider mixed precision,
    - ...

Conclusions

Final remarks

Conclusions:

- Contribution to parallel and efficient implementation of HMSC-R package.
- Evaluation of performance using accelerators on Puhti
  - 6–70x speedup compared to sequential R implementation.

Future perspectives:

- **Distributed compute** with TensorFlow
  - tf.distribute.Strategy is a TensorFlow API that implements distributed training
  - Easy to use and switching between strategies:
    - MirroredStrategy and MultiWorkerMirroredStrategy
    - ParameterServerStrategy
- Experiment with **mixed precision**
  - FP16, FP32, FP64
  - BFloat16, TF32

Synthesis by Gleb

1. Ultimate goal — faster analysis of species communities
   - Change model
   - Change fitting approach
   - Optimize numerical routines
   - Use more prominent backend
2. Porting block-Gibbs of Hmsc-R to GPU-compatible code
   - Almost full functionality of Hmsc-R is re-implemented in Python+TF
   - 2200 lines of sampler R code $\longrightarrow$ 1300 lines of Python+TF
   - Many loops replaced with vectorized computation
   - Utilities for export and import from R interface to Python+TF sampler
3. Validating the new implementation — lot of small errors!
4. Computational performance assessment
   - Up to $\times 30$ speed-up recorded. Can we get more?
   - Different speed-up in different models — what affects?
   - NNGP and Polya-Gamma rely on non-TF libraries
   - Profiling with TensorBoard to identify untrivial bottlenecks
5. Tuning the code for efficiency on AMD GPU in Lumi-G
6. Developing a proper (automatic) code testing set-up

Perspectives for further development by Gleb

1. Porting HMSC features that are yet in developmental phase
    - Spatio-temporal modelling
    - Structured increases shrinkage
2. Parallel multi-GPU computing beyond independent Markov chains
    - Shorter but numerous chains?
    - Within chain multi-GPU utilization
    - Non-MCMC Bayesian fitting
3. Mixed precision
4. Design hybrid sampler
    - TensorFlow enables autodiff for free
    - Identify the typical bottlenecks of autocorrelation in Gibbs sampler
    - Augment the scheme with non-conjugate conditional samplers