Efficient simulations of spiking neural networks using NEST GPU

CNS 2025 Software showcase

José Villamar[1,2], **Luca Sergi**[3,4], Tiddia G.[4], Babu P.[5], Lonardo A.[6], Simula F.[6], Pontisso L.[6], Pastorelli E.[6], Paolucci P.S.[6], Golosio B.[3,4], Senk J[1,7].

1. Institute for Advanced Simulation (IAS-6) Jülich Research Centre, Jülich, Germany
2. RWTH Aachen University, Aachen, Germany
3. Dipartimento di Fisica, Università di Cagliari, Monserrato, Italy
4. Istituto Nazionale di Fisica Nucleare, Sezione di Cagliari, Monserrato, Italy
5. Simulation and Data Laboratory Neuroscience, Jülich Supercomputing Centre, Jülich Research Centre, Jülich, Germany
6. Istituto Nazionale di Fisica Nucleare, Sezione di Roma, Roma, Italy
7. Sussex AI, School of Engineering and Informatics, University of Sussex, Brighton, United Kingdom

j.villamar@fz-juelich.de and lsergi@dsf.unica.it

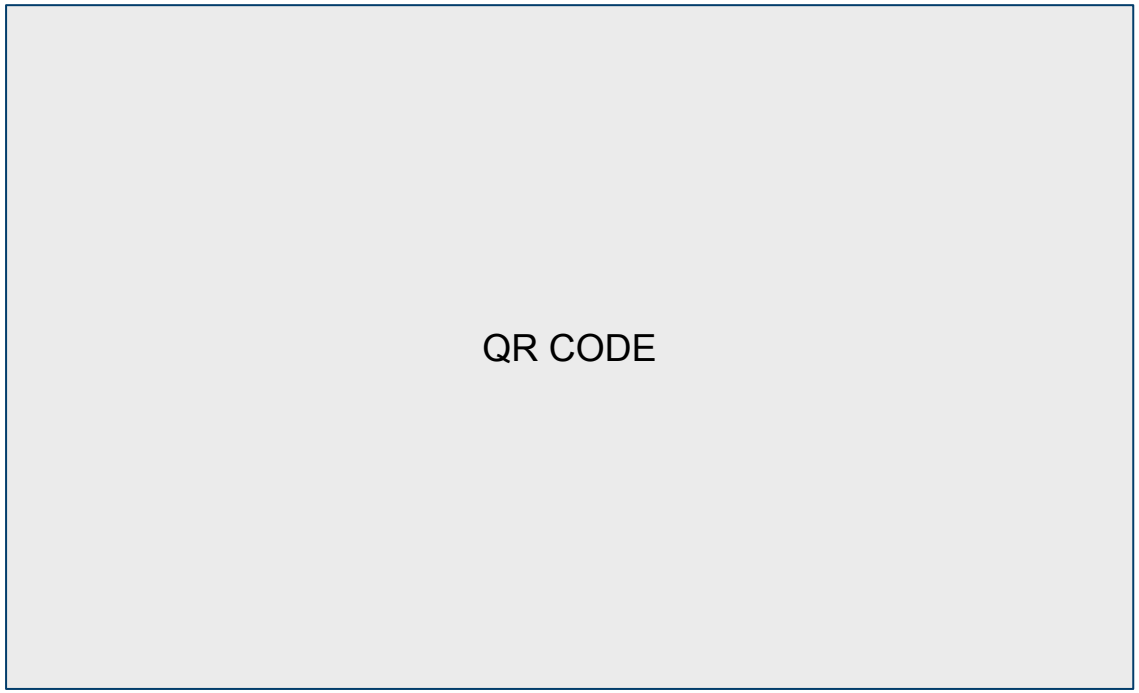
Member of the Helmholtz Association

JÜLICH Forschungszentrum

First things first

Material used during the showcase is available at

Zenodo link



Outline

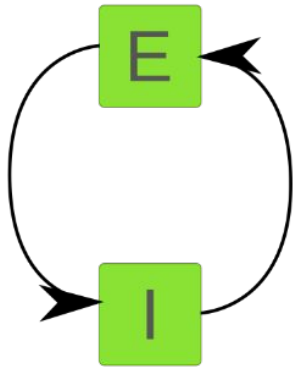
- What is NEST and what is NEST GPU?
  - Our role in the NEST initiative
- Modeling your first network with NEST GPU
  - An example implementation using the Brunel network
- First steps on spike data analysis
  - Example using first and second order statistics
- Scaling up your network on multiple GPUs
  - A sneak peek on large scale spiking neural network simulations
- Closing remarks
  - Related works and future plans with NEST GPU

Outline

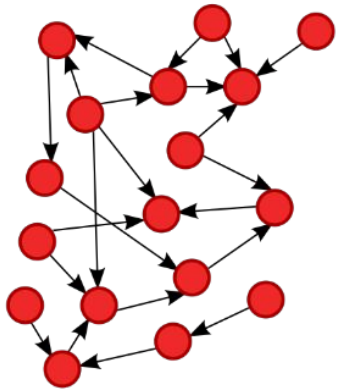
- What is NEST and what is NEST GPU?
  - Our role in the NEST initiative
- Modeling your first network with NEST GPU
  - An example implementation using the Brunel network
- First steps on spike data analysis
  - Example using first and second order statistics
- Scaling up your network on multiple GPUs
  - A sneak peek on large scale spiking neural network simulations
- Closing remarks
  - Related works and future plans with NEST GPU

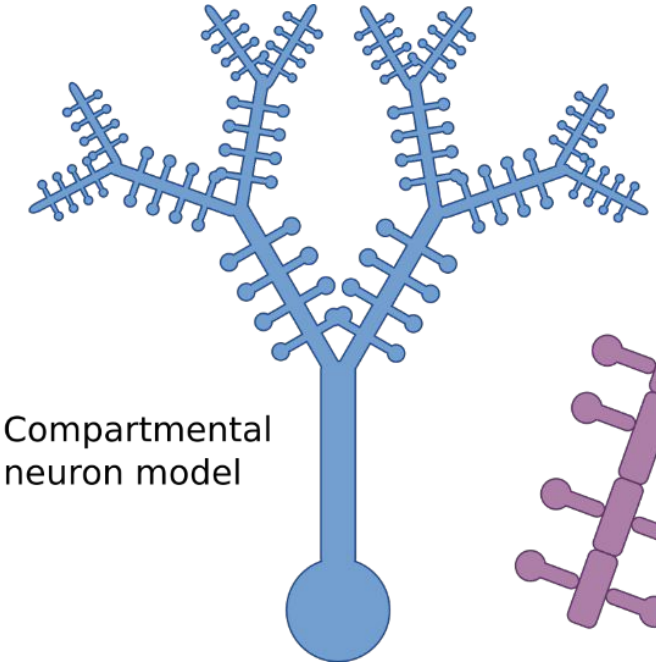
Different solutions for problems with different sizes

Population model



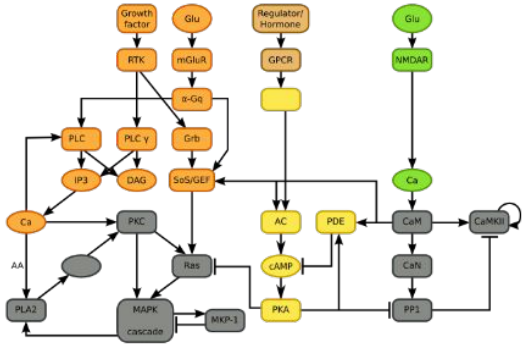
Point neuron network model



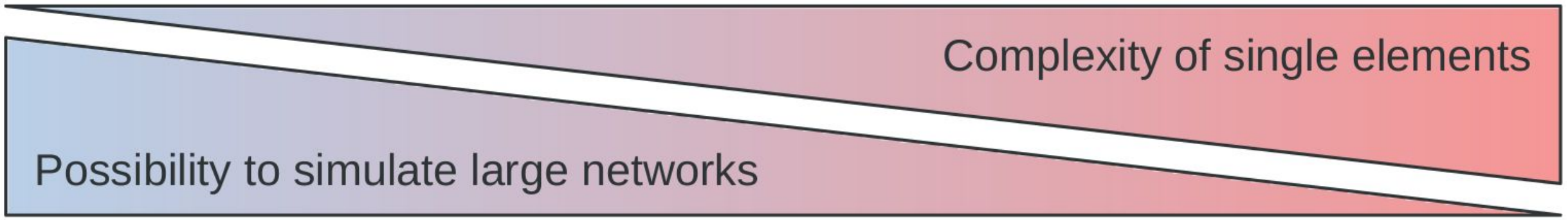


Compartmental neuron model

Compartmental membrane model

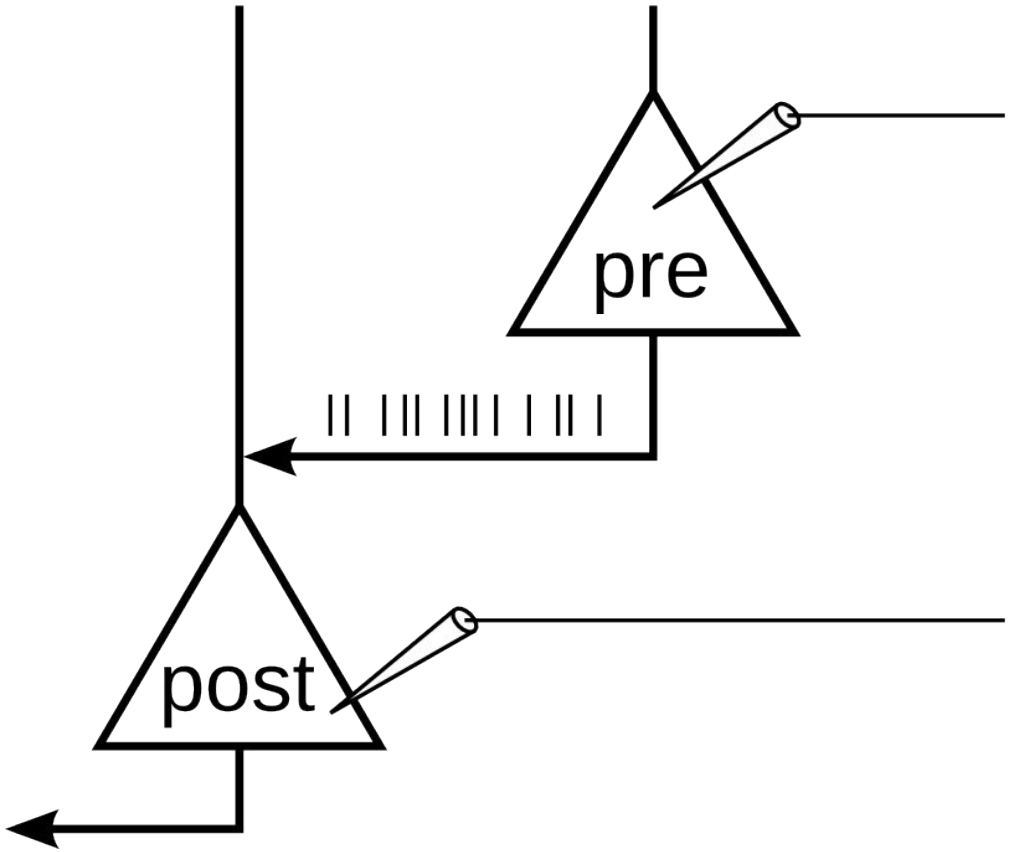


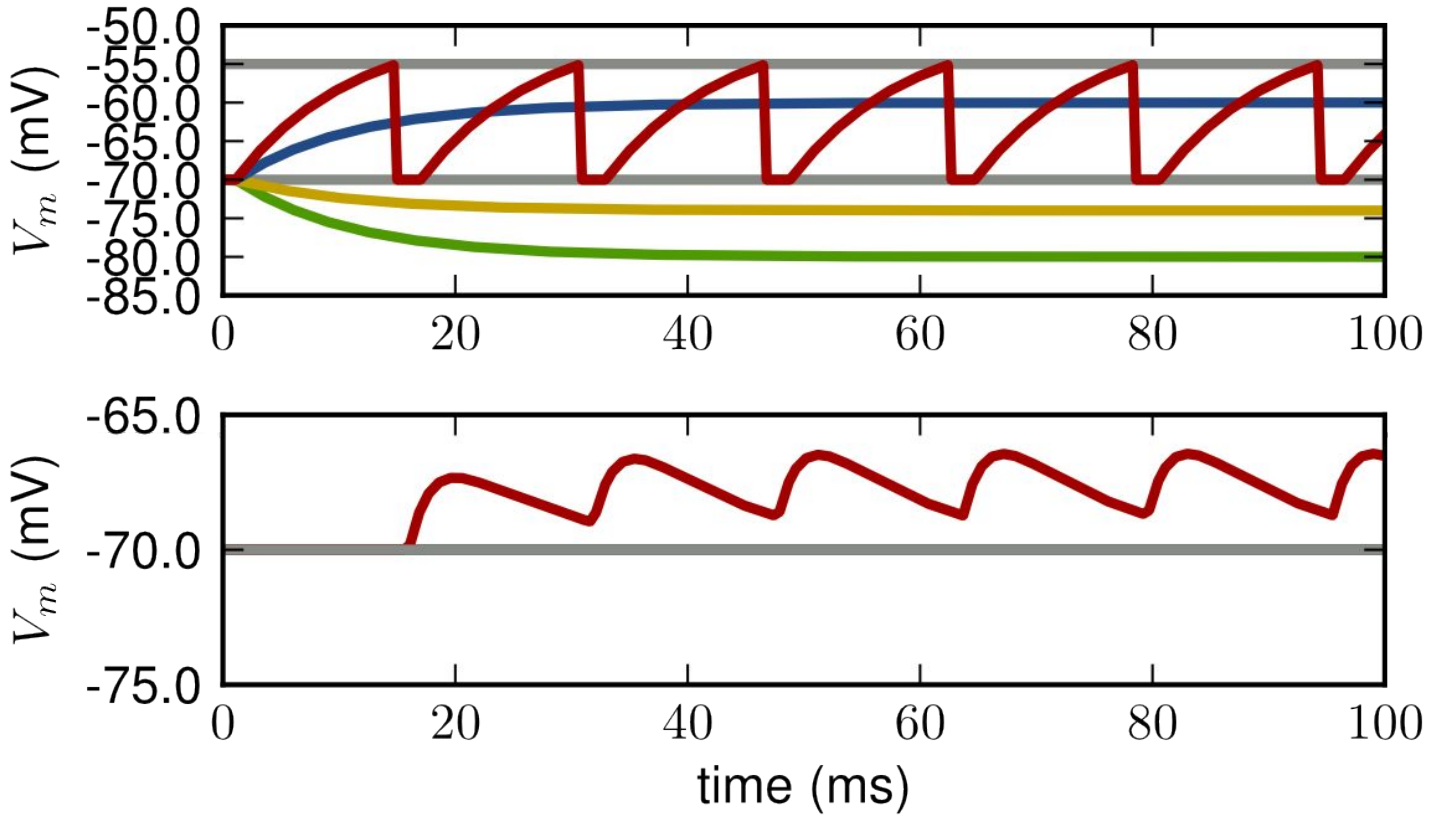
Reaction-diffusion model



Different solutions for problems with different sizes

Point neuron simulations mimic a neuroscientific experiment





NEST == NEURAL SIMULATION TOOL

- C++ core, hybrid parallelization (OpenMP+MPI), Python frontend PyNEST
- Same code from laptops to supercomputers → simulation of large-scale models
- Development is driven by scientific needs with a focus on accuracy and flexibility as well as quality assurance

Main website:	https://www.nest-simulator.org
NEST initiative:	https://www.nest-initiative.org
Source code:	https://github.com/nest/nest-simulator
Online documentation:	https://nest-simulator.readthedocs.io
EBRAINS:	https://ebrains.eu/service/nest-simulator

What is NEST GPU?

NEST GPU as the GPU backend for the NEST simulator

- CUDA kernels for state update and multi-GPU support using MPI for spike communication
- Similar Python interface
- Many neuron models already implemented, e.g., IAF, conductance and current based, Izhikevich
- Synapse models: standard synapse, nearest-neighbor STDP

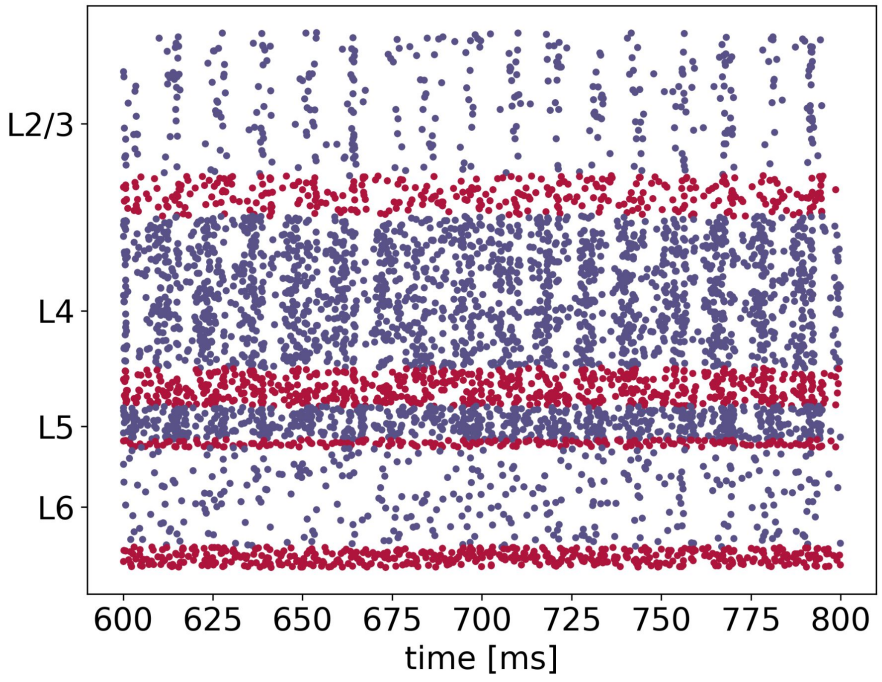
nest::gpu

https://github.com/nest/nest-gpu

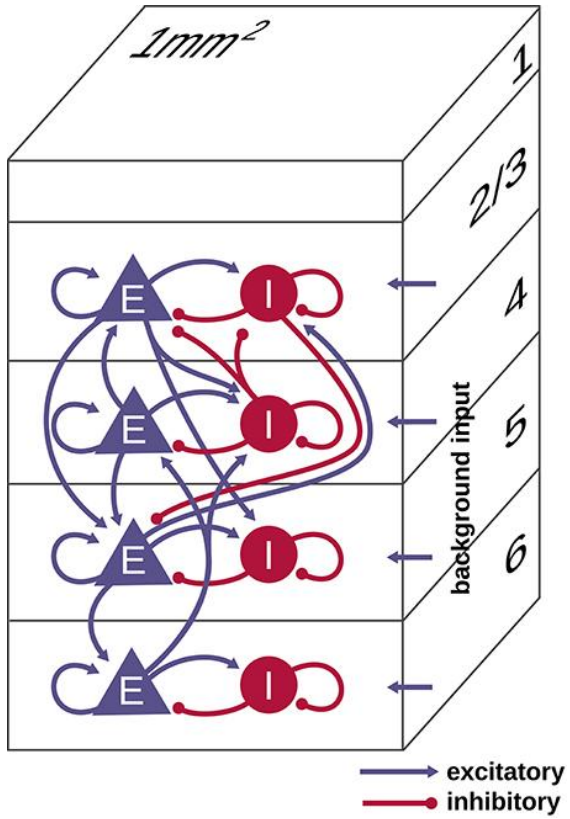
Have a look at the documentation!

Are simulation results compared to NEST CPU the same?

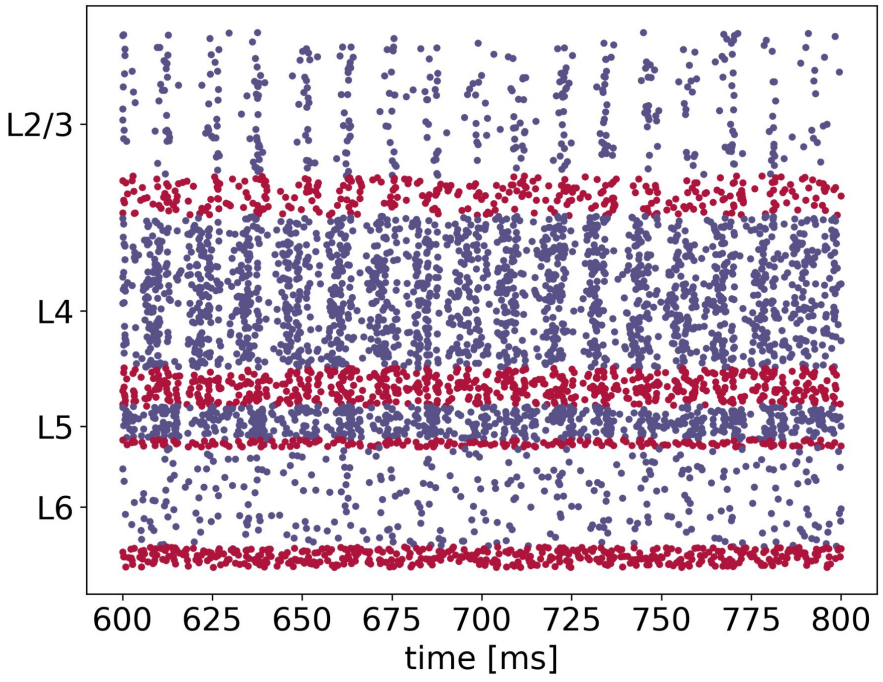
Spiking activity is not identical but statistically equivalent. Golosio et al. (2021)



NEST GPU raster plot of the microcircuit model



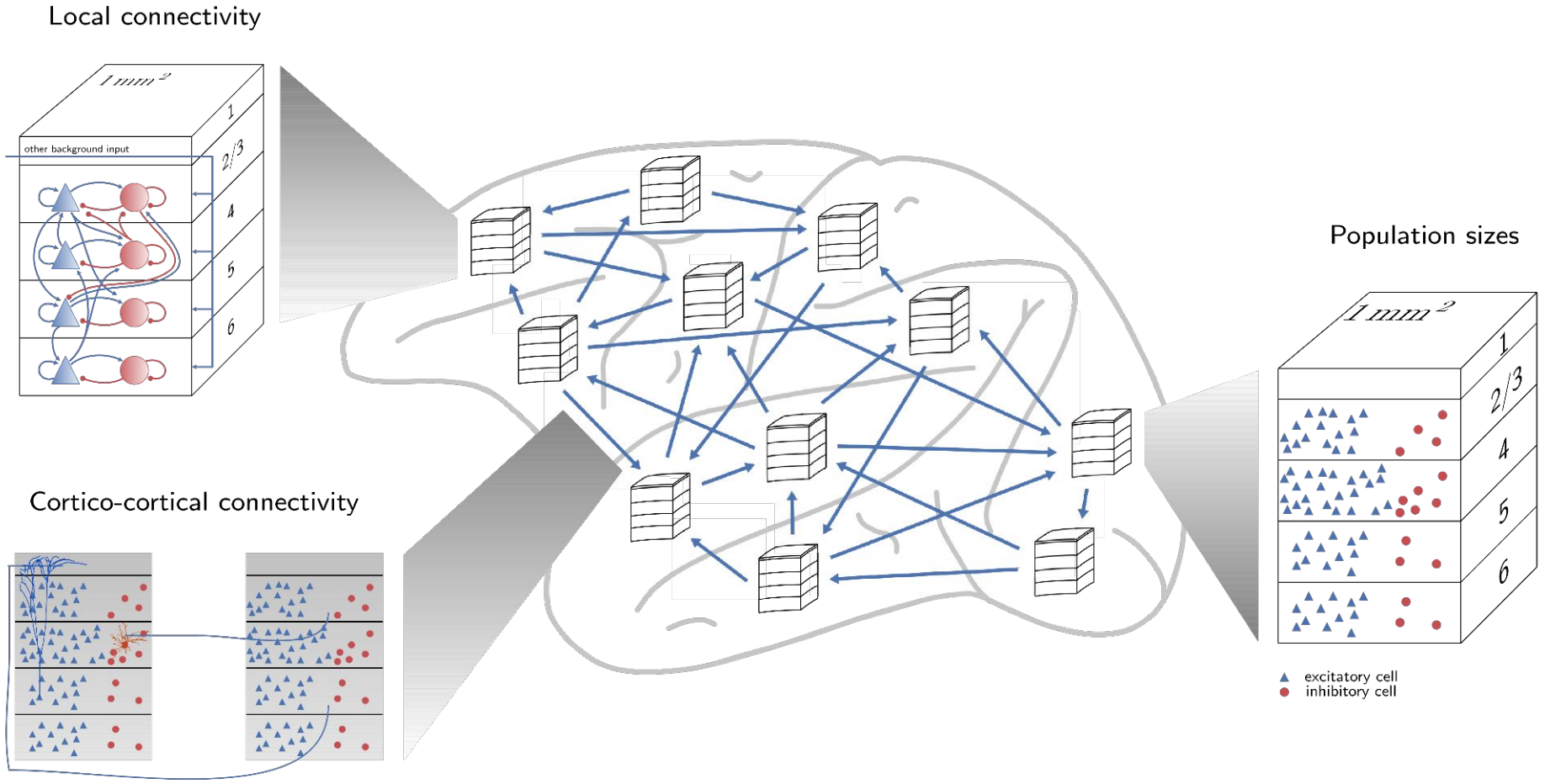
Schematic illustration of the Potjans & Diesmann microcircuit model (2014). Adapted from van Albada et al. (2018)



NEST raster plot of the microcircuit model

Support for larger network models

Multi-area model of the macaque visual cortex simulated on a multi-GPU cluster. Tiddia et al. (2022)



Schematic illustration of the multi-area model.
Adapted from Schmidt et al. (2018)

First steps with NEST GPU

Speaker: Luca Sergi

- What is NEST and what is NEST GPU?
  - Our role in the NEST initiative
- Modeling your first network with NEST GPU
  - An example implementation using the Brunel network
- First steps on spike data analysis
  - Example using first and second order statistics
- Scaling up your network on multiple GPUs
  - A sneak peek on large scale spiking neural network simulations
- Closing remarks
  - Related works and future plans with NEST GPU

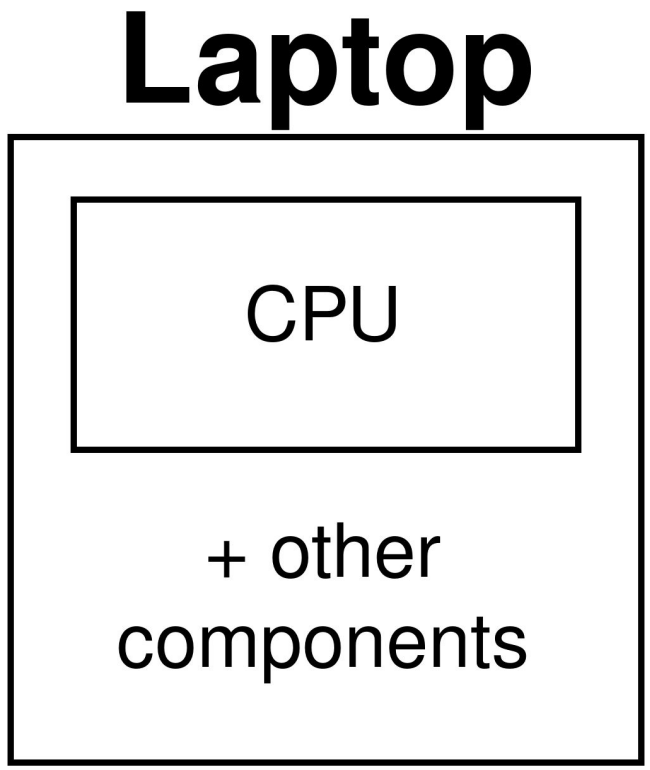
Going large scale with NEST GPU

Speaker: José Villamar

- What is NEST and what is NEST GPU?
  - Our role in the NEST initiative
- Modeling your first network with NEST GPU
  - An example implementation using the Brunel network
- First steps on spike data analysis
  - Example using first and second order statistics
- Scaling up your network on multiple GPUs
  - A sneak peek on large scale spiking neural network simulations
- Closing remarks
  - Related works and future plans with NEST GPU

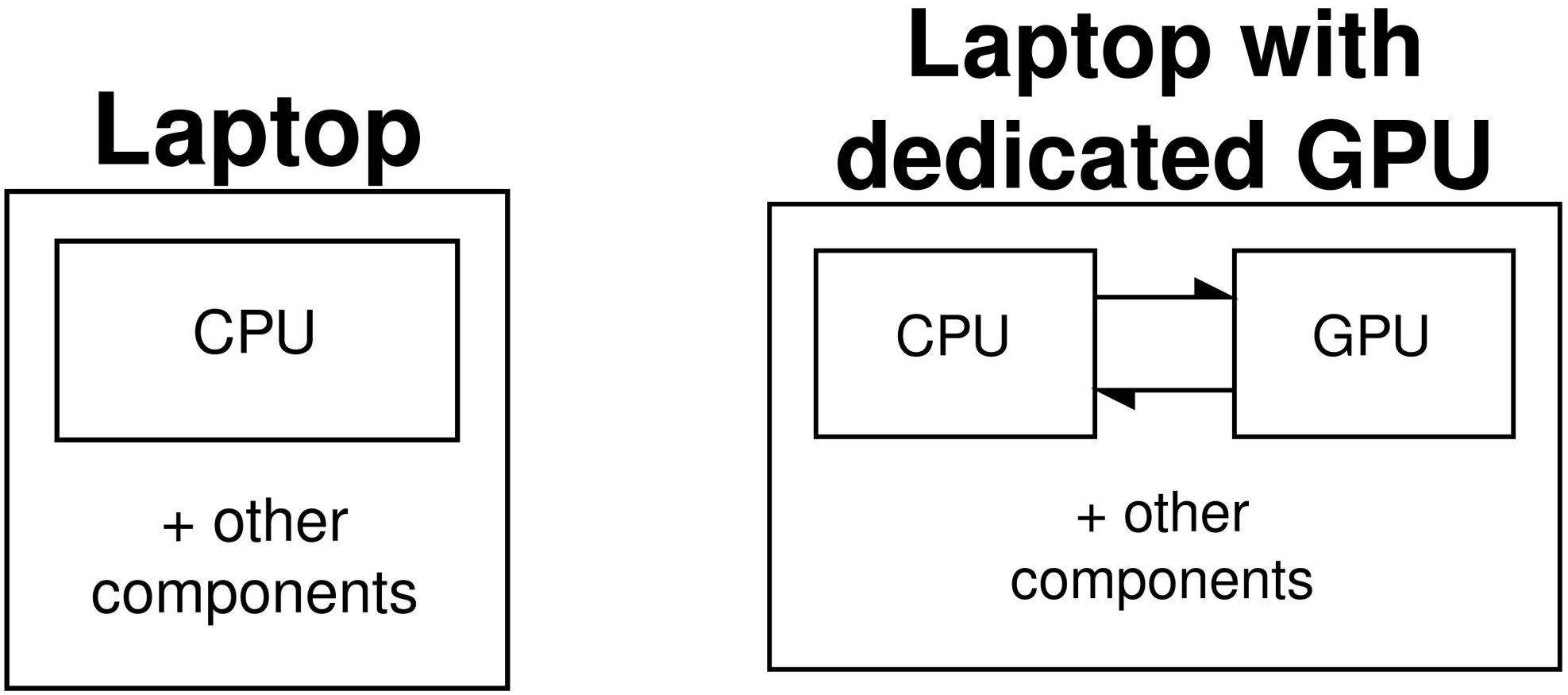
A deeper look into computing hardware

From laptops to clusters



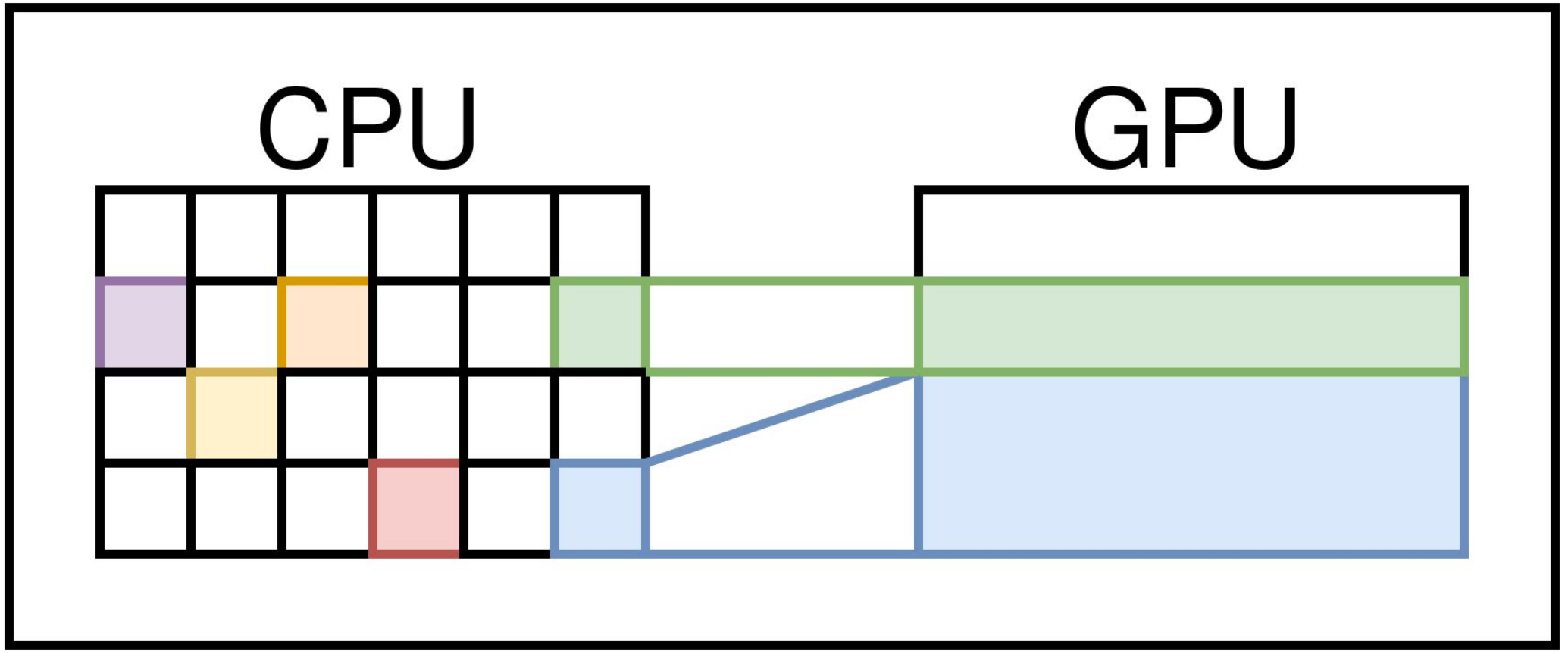
A deeper look into computing hardware

From laptops to clusters



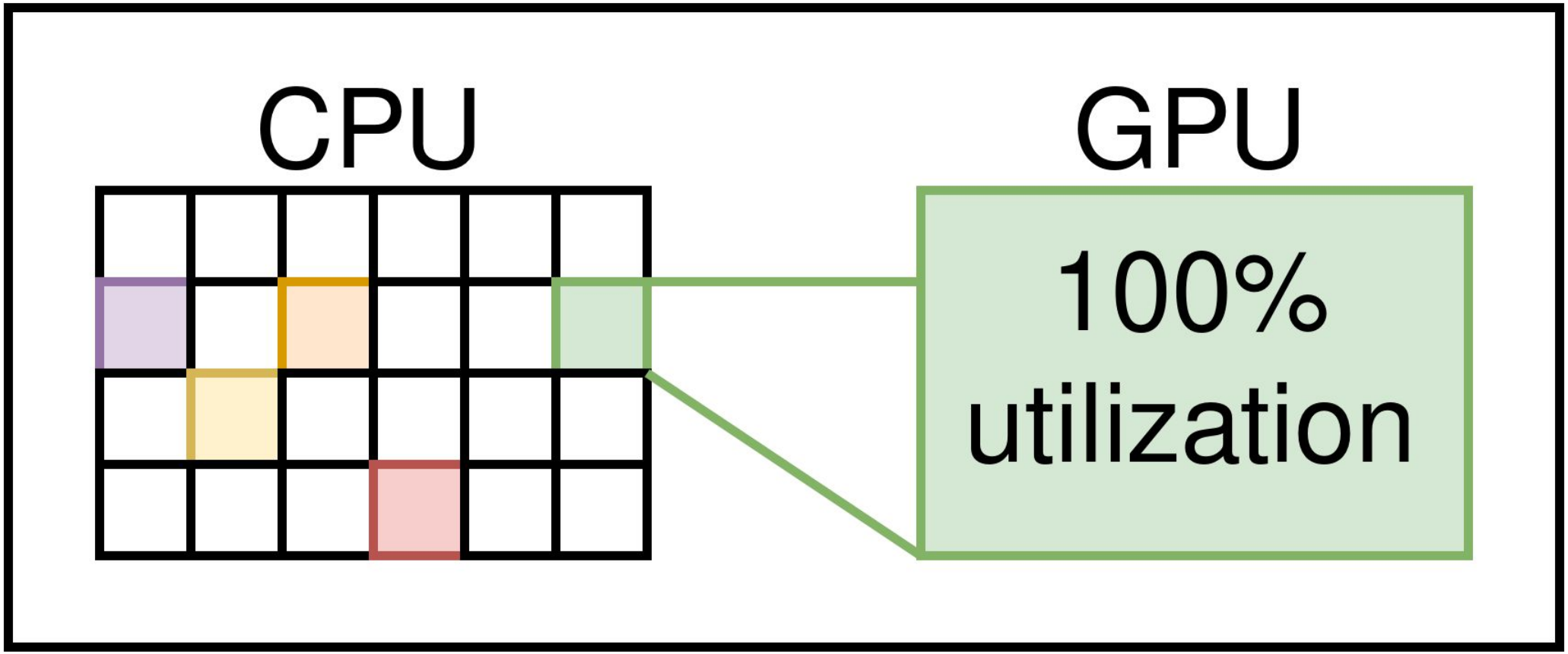
A deeper look into computing hardware

From laptops to clusters



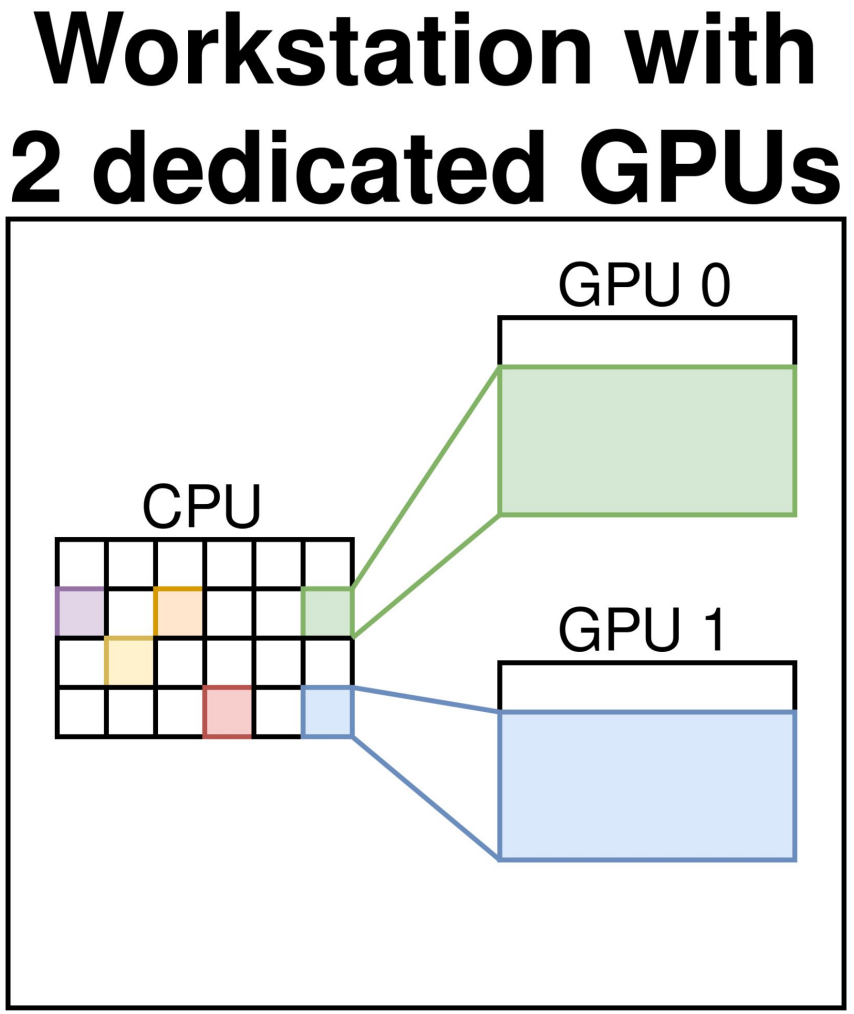
A deeper look into computing hardware

From laptops to clusters



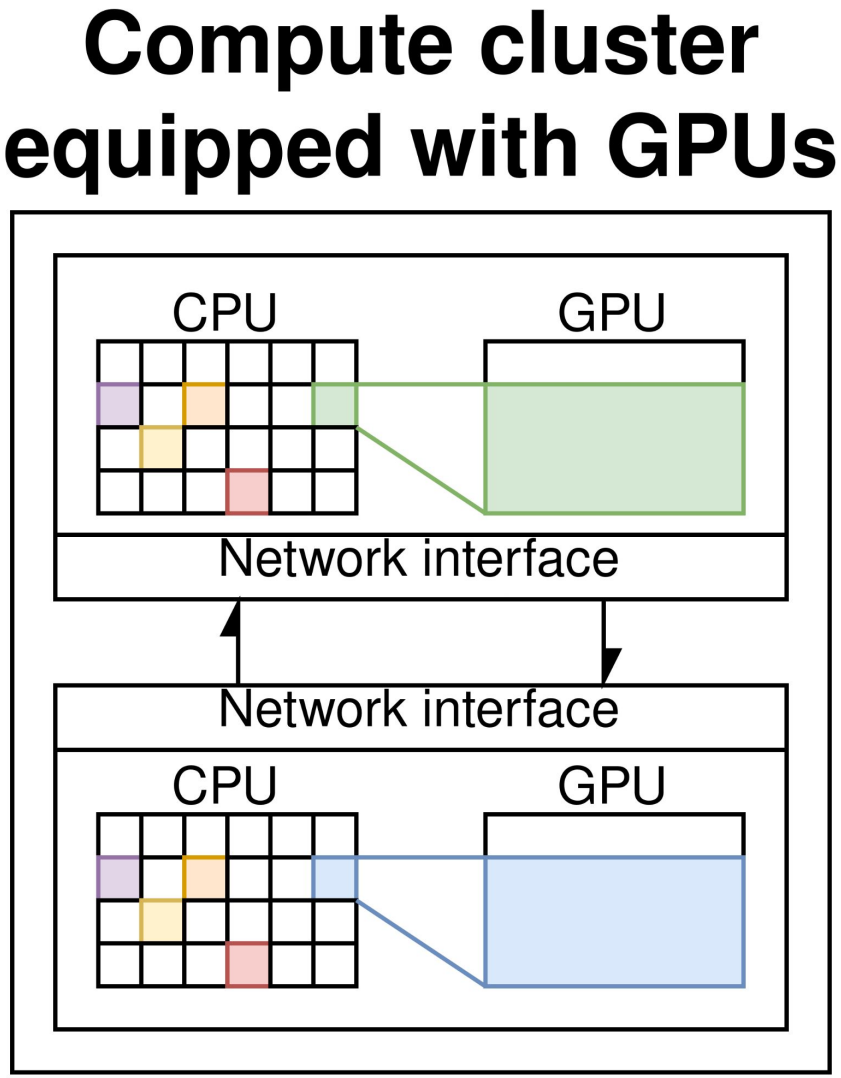
A deeper look into computing hardware

From laptops to clusters



A deeper look into computing hardware

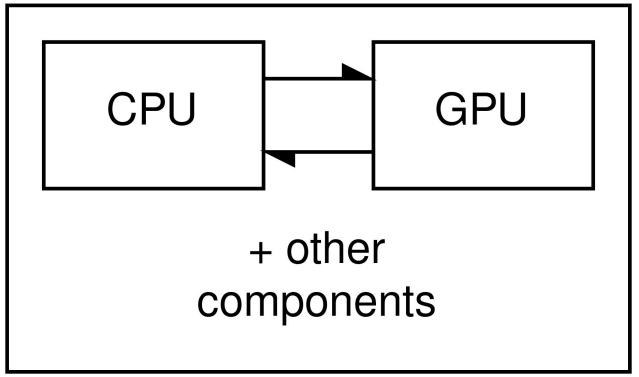
From laptops to clusters



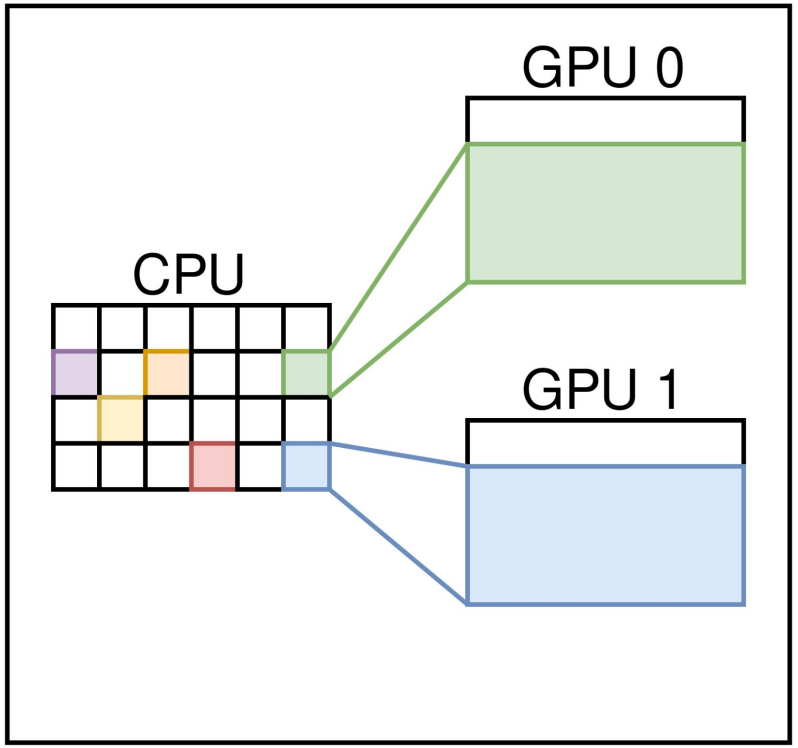
A deeper look into computing hardware

From laptops to clusters

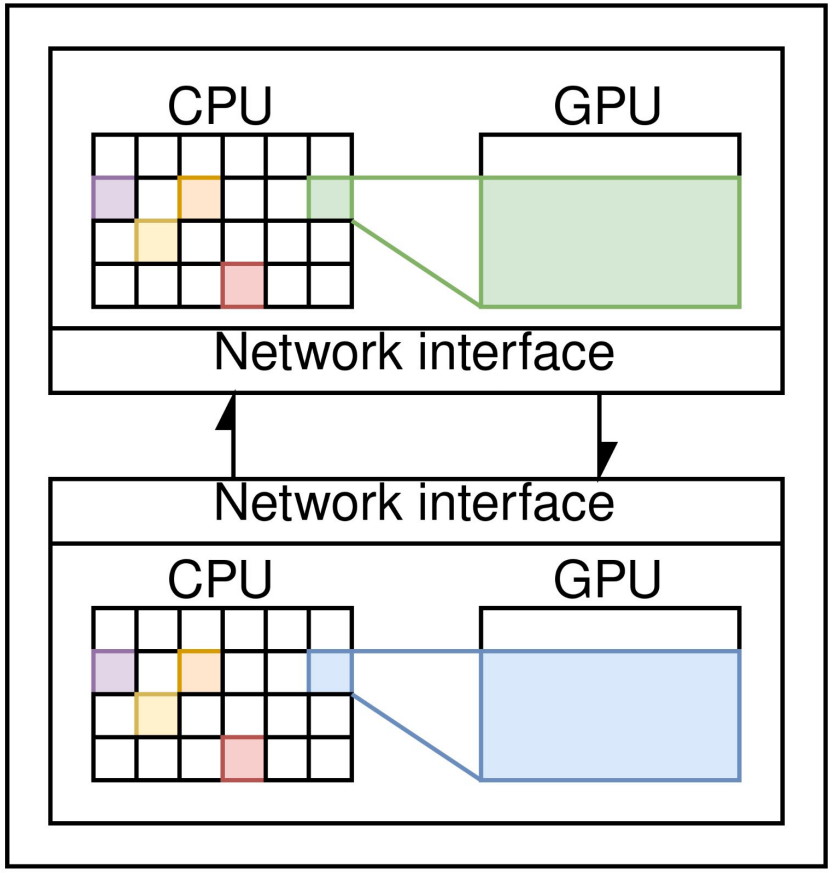
Laptop with dedicated GPU



Workstation with 2 dedicated GPUs



Compute cluster equipped with GPUs



Q: Why does the computing architecture matter?

A: It shouldn't, we take care of it by parallelizing through MPI :)

- The interface of NEST GPU does not globally control all GPUs at once
- The user needs to specify which parts of the network go to each GPU
- Each GPU is controlled by a single MPI process
- The interface is designed so that users define networks in an MPI aware structure

MPI specific interface functions

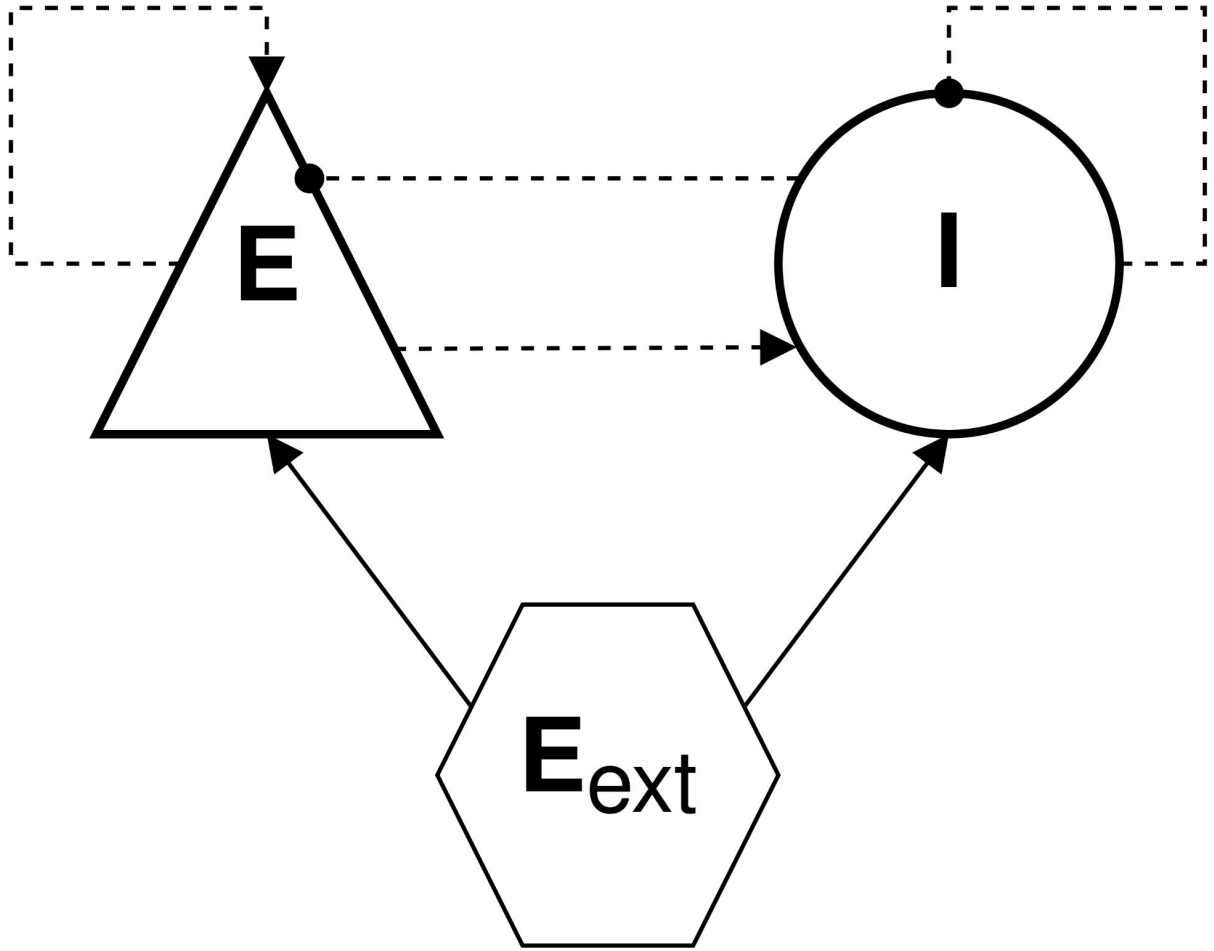
Users instantiate networks either locally without MPI interaction or "remotely" through MPI coordination:

```
nestgpu.Create( neuron model, neuron count, *params )
nestgpu.Connect( source neuron population, target neuron population,
                 *params )
```

```
nestgpu.RemoteCreate( MPI rank, neuron model, neuron count, *params )
nestgpu.RemoteConnect( source MPI rank, source neuron population,
                       target MPI rank, target neuron population, *params )
```

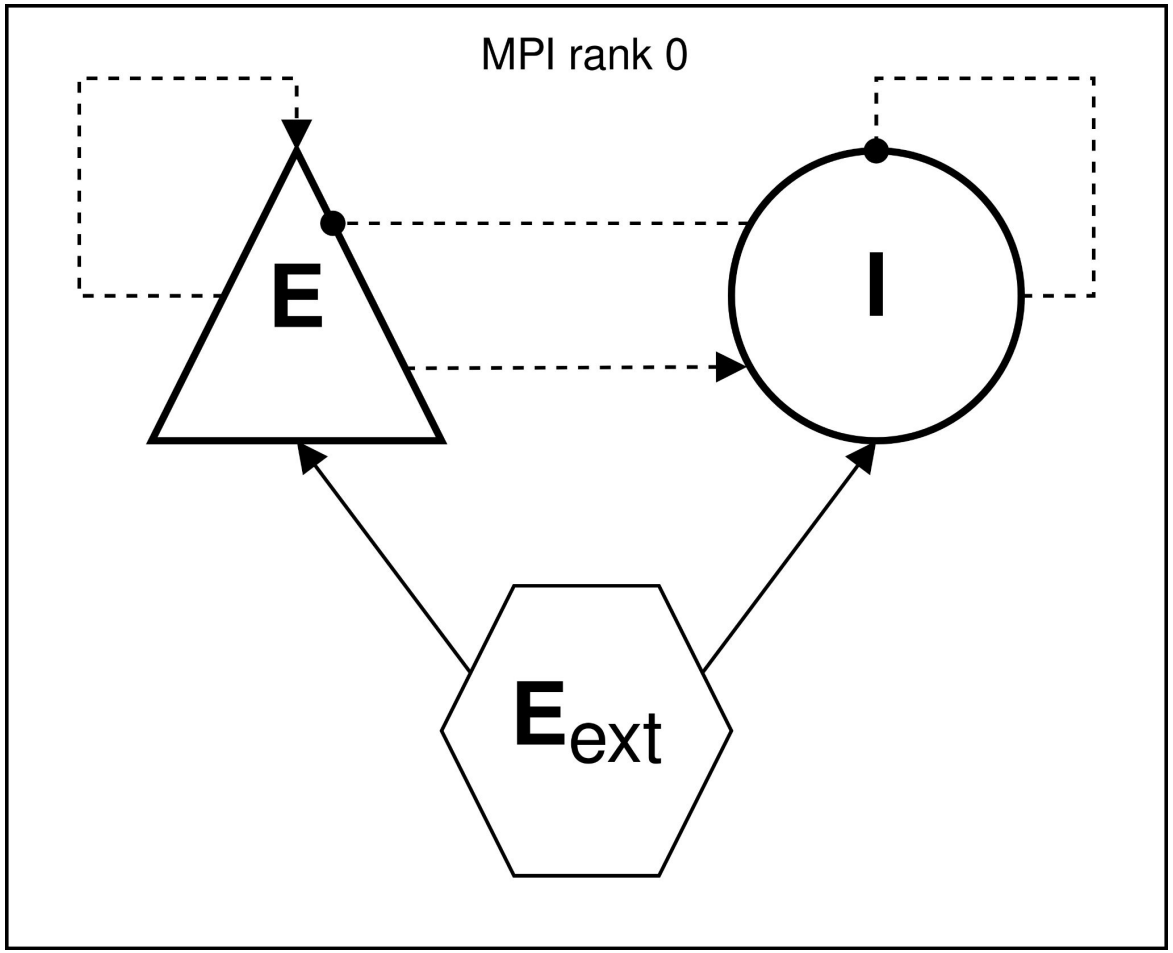
Putting things in practice

Starting from the previously viewed Brunel network



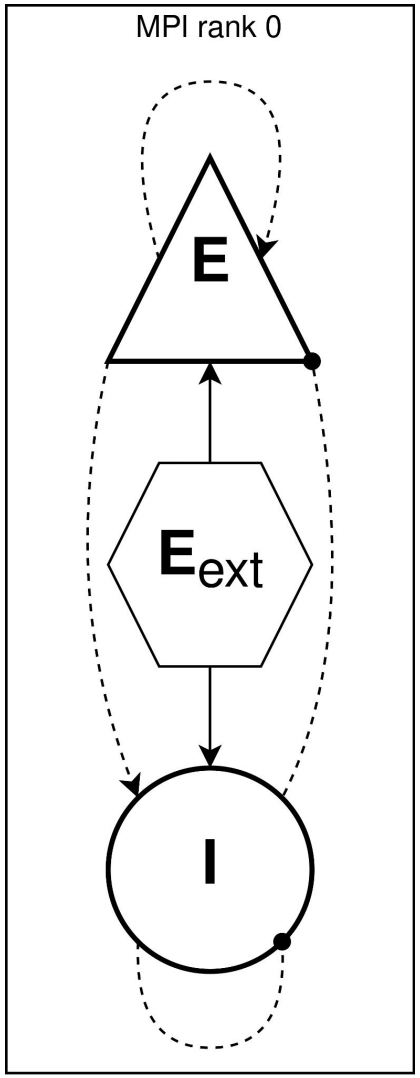
Putting things in practice

Starting from the previously viewed Brunel network



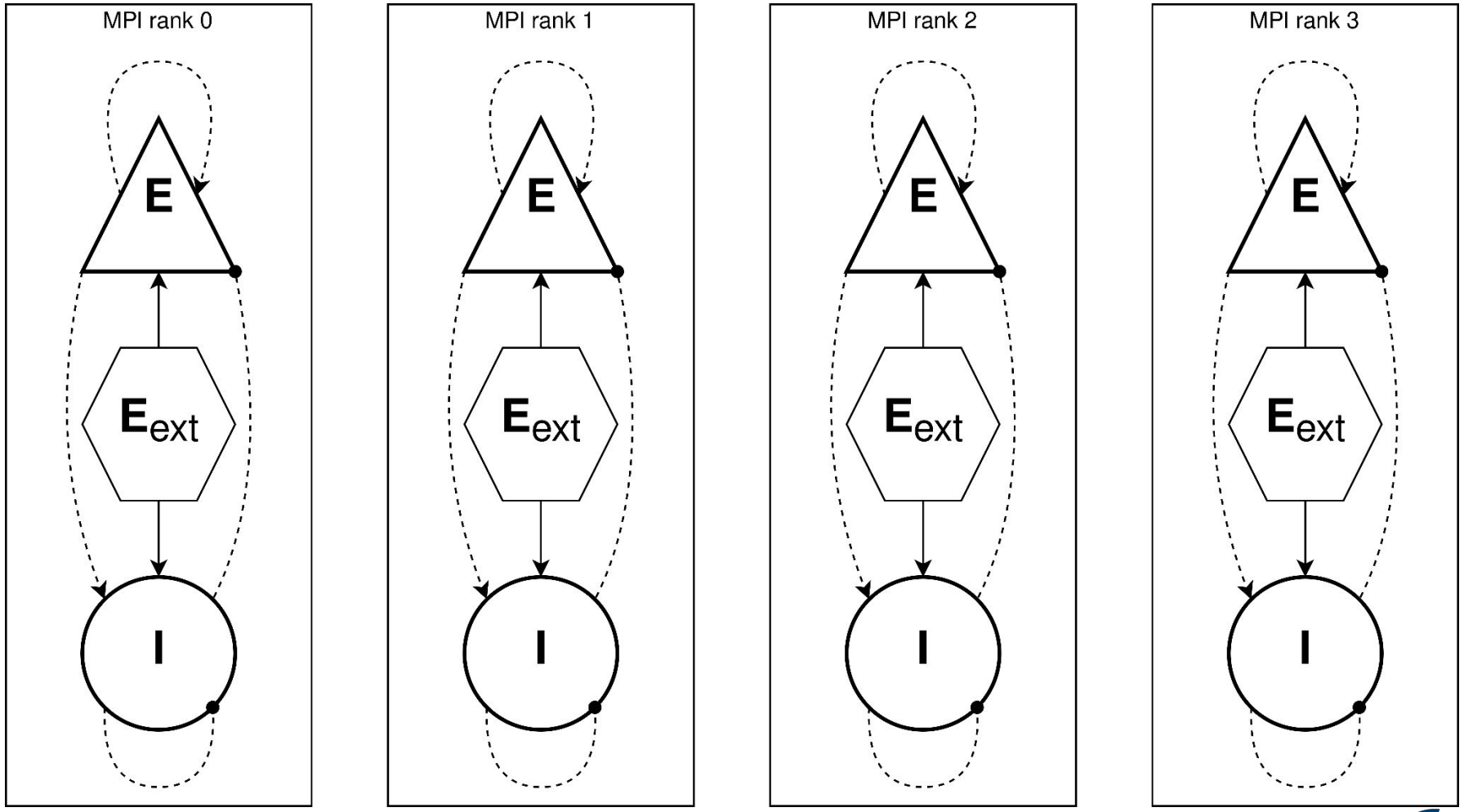
Putting things in practice

Starting from the previously viewed Brunel network



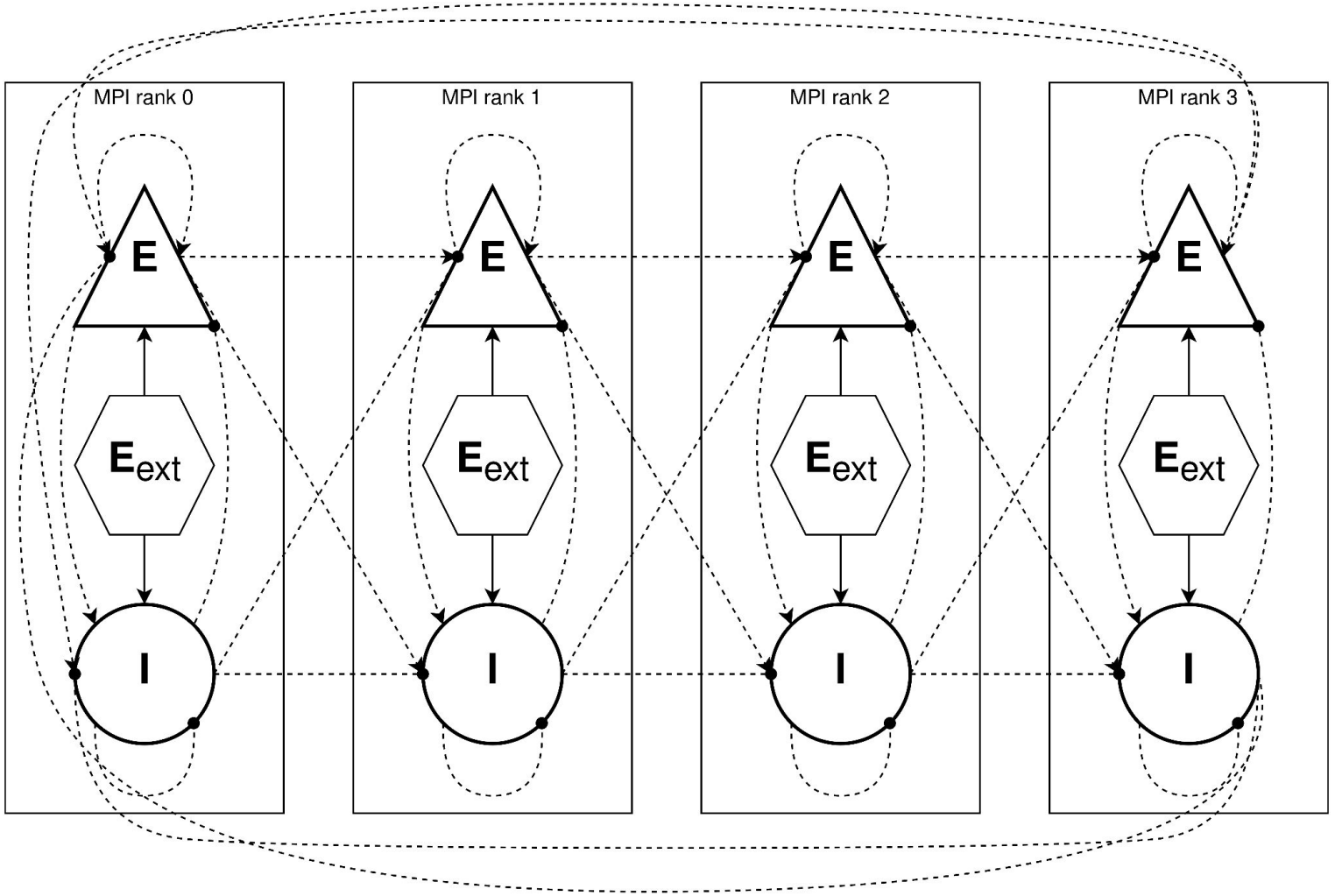
Putting things in practice

Going to multiple Brunel networks across MPI processes



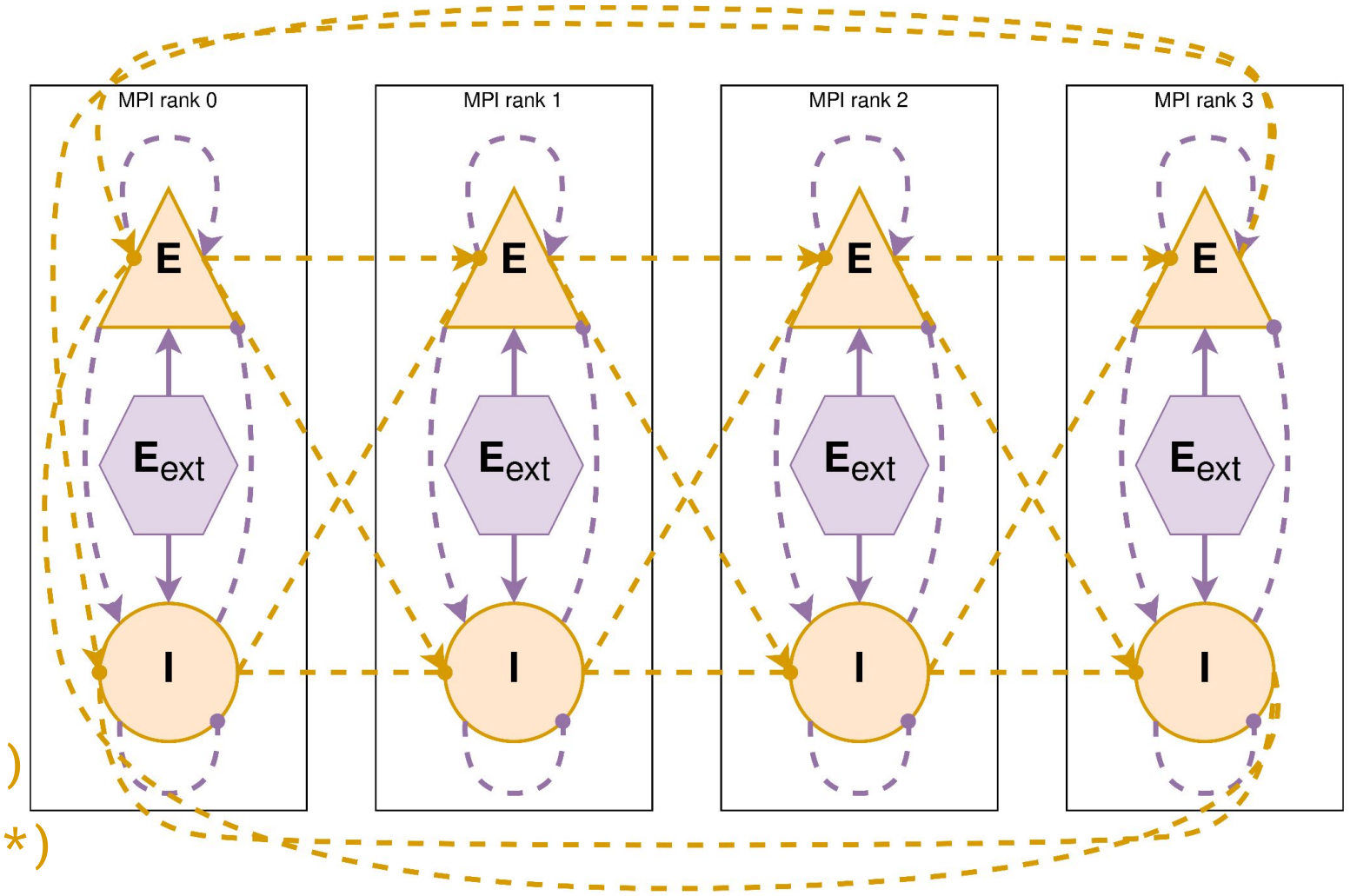
Putting things in practice

Interconnecting each Brunel network in a feed-forward ring topology



Putting things in practice

Interconnecting each Brunel network in a feed-forward ring topology



`Create(*)`

`Connect(*)`

`RemoteCreate(*)`

`RemoteConnect(*)`

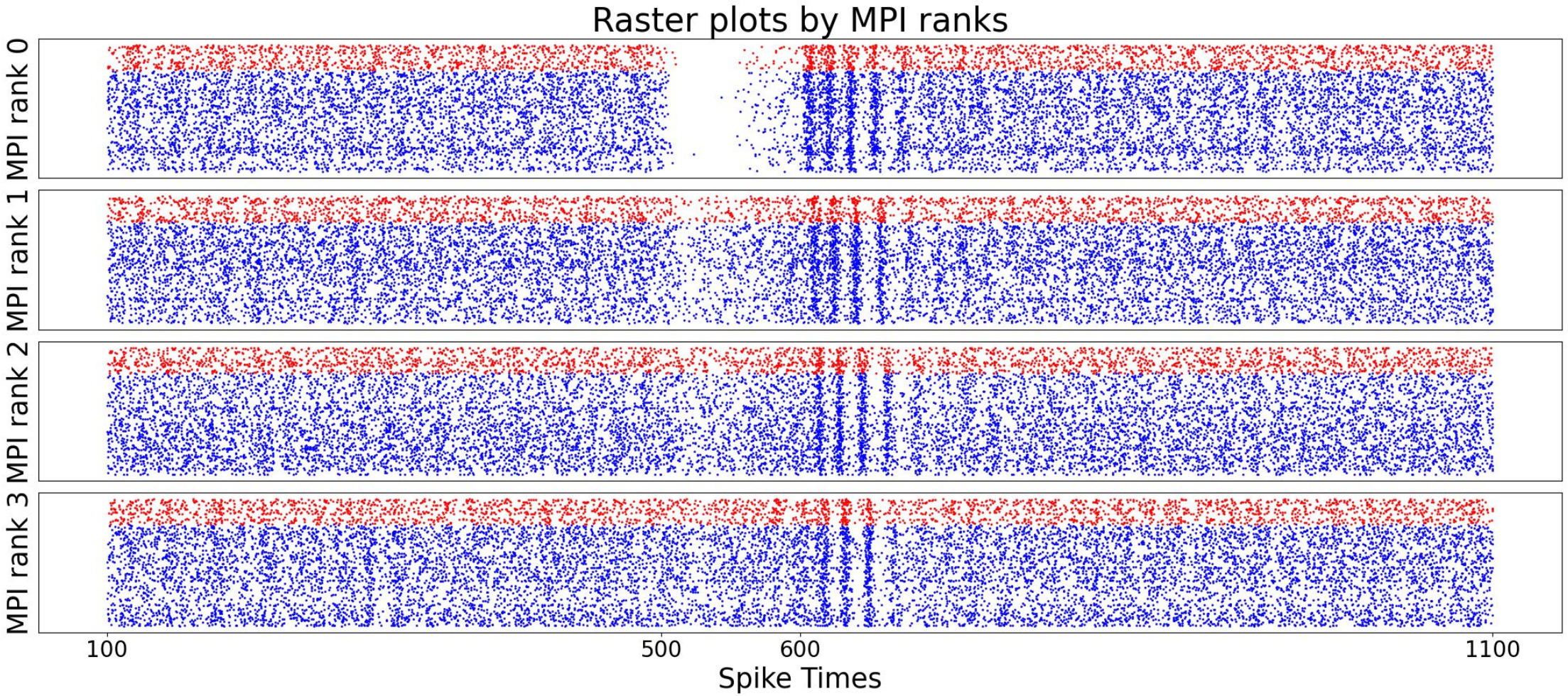
Putting things in practice

Dummy experiment with Brunel ring network

- Each GPU (individual MPI process) houses an instance of the Brunel network
- Each network instance is driven by an individual Poisson generator
- Total number of incoming connections per neurons remain constant regardless of the number of network instances
- Incoming connections are split into remote sources and local sources
  - 60% of excitatory connections come from remote sources
  - 10% of inhibitory connections come from remote sources
- Each network instance receives connections from a neighbor instance and sends connections to another instance
  - By creating connections between "left" and "right" neighbors, all instances form a feed-forward ring topology
- After 500ms the rate of a single Poisson generator is reduced by 25% for 100 ms

Putting things in practice

And getting some interesting(?) results



Going large scale with NEST GPU

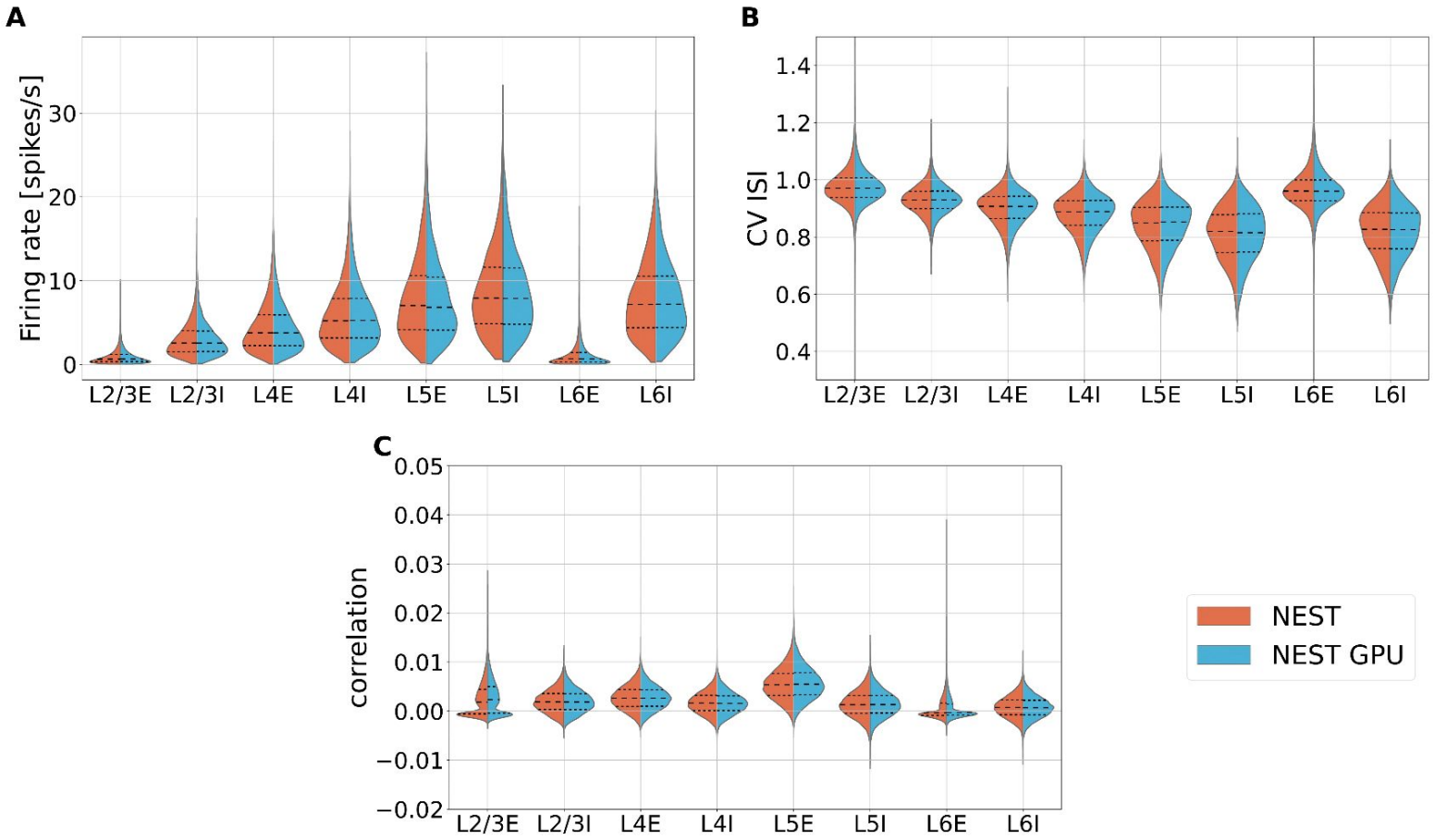
Speaker: José Villamar

- What is NEST and what is NEST GPU?
  - Our role in the NEST initiative
- Modeling your first network with NEST GPU
  - An example implementation using the Brunel network
- First steps on spike data analysis
  - Example using first and second order statistics
- Scaling up your network on multiple GPUs
  - A sneak peek on large scale spiking neural network simulations
- Closing remarks
  - Related works and future plans with NEST GPU

Related works

- Simulation of Potjans & Diesmann microcircuit model on a single GPU
  - Golosio et al. (2021) 10.3389/fncom.2021.627620
- Multi-area model of the macaque visual cortex simulated on a multi-GPU cluster
  - Tiddia et al. (2022) 10.3389/fninf.2022.883333
- Improvements on network construction methods
  - Golosio, Villamar, Tiddia et al. (2023) on a single GPU 10.3390/app13179598
  - Golosio, Villamar, Tiddia et al. (in preparation) on multiple GPUs

Related works: Spiking data comparison using PD14

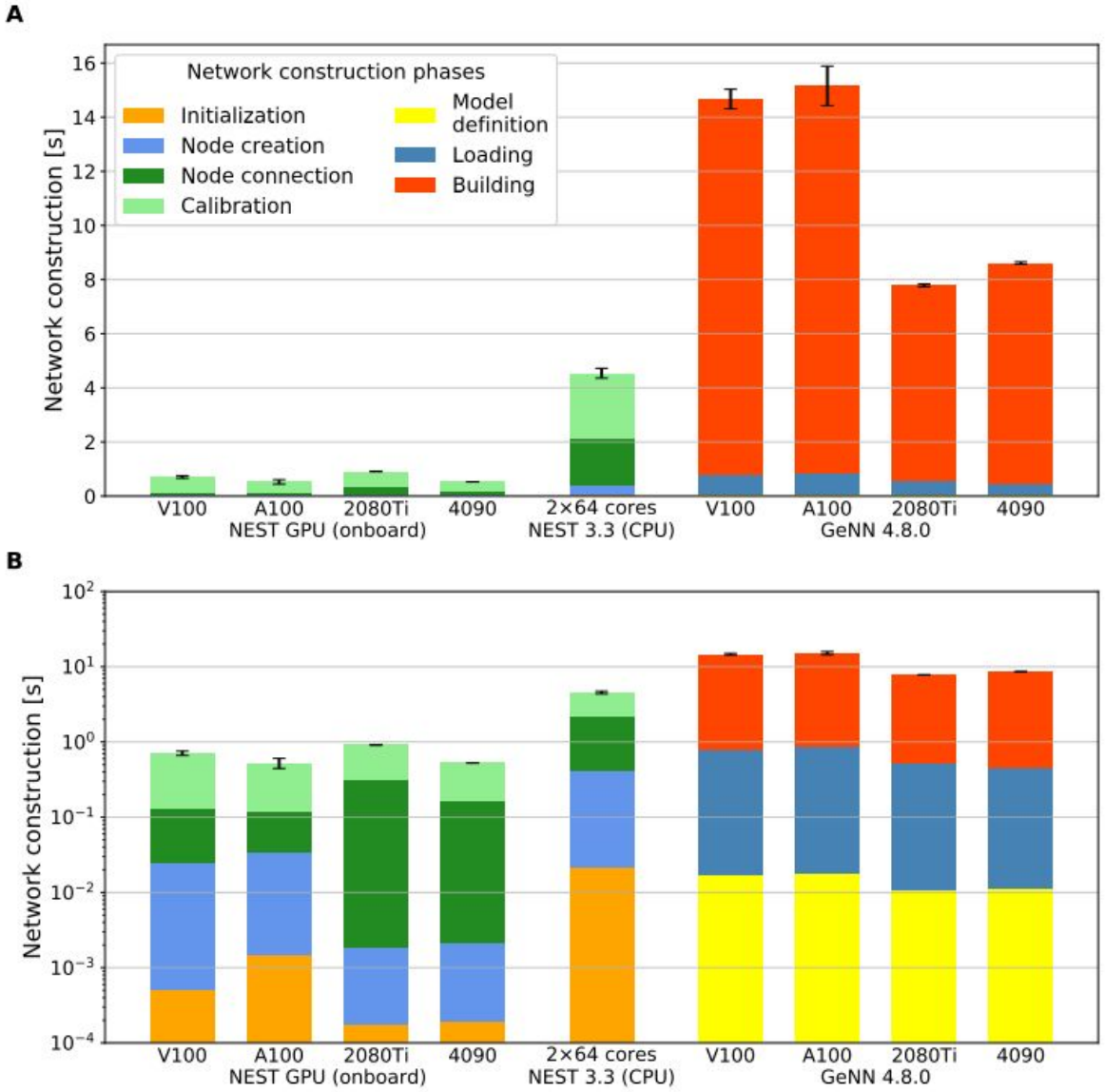


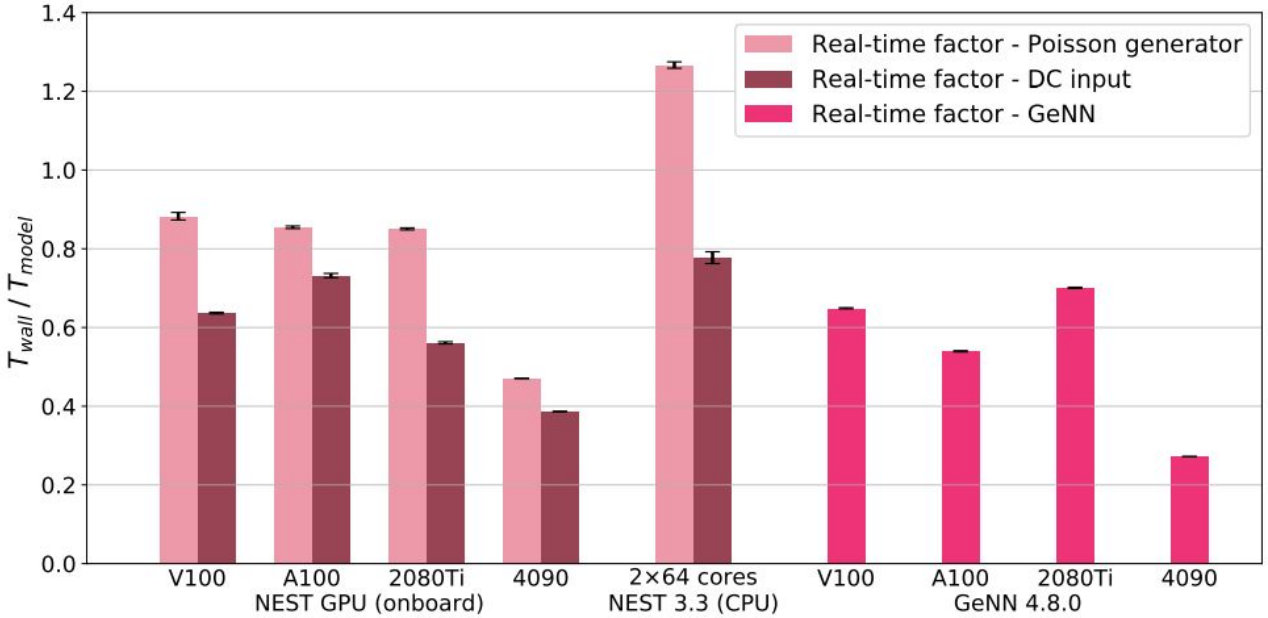
Violin plots of the distributions of **firing rate** (**A**), **CV ISI** (**B**), and **Pearson correlation** (**C**) of simulations of the populations of the microcircuit model using NEST GPU and NEST 3.3.

Spiking activity was recorded for 600s of simulation time after 500ms of pre-simulation time using a resolution of 0.1ms.

Golosio, Villamar, Tiddia et al. (2023)

Related works: Performance comparison using PD14





- Golosio, Villamar, Tiddia et al. (2023)
- Network construction in ~0.5 s
- Real-time factor:
  - Poisson generator: ~0.5
  - DC input: ~0.4

Future plans with NEST GPU

- Kernel optimizations:
  - Optimize single GPU performance
  - Use hardware topology to define machine aware MPI neighborhoods
- Code base alignment:
  - Compatibility with NESTML to expand upon available neuron and synapse models
  - Unify Python interface for easier user interaction between CPU and GPU backends
  - Bring more of the available features of the CPU simulator
    - Next planned feature is spatially defined networks
  - Merge simulation kernel backends to achieve a single code base

Q&A