Preparing for exascale computing: Large-scale neuronal network construction through parallel GPU memory instantiation

Villamar, J.[1,2], Golosio, B.[3,4], Tiddia, G.[3,4], Sergi, L.[3,4], Pontisso, L.[5], Simula, F.[5], Lonardo, A.[5], Pastorelli, E.[5], Paolucci, P.S.[5], Diesmann, M.[1,6,7], Morrison, A.[1,8], Senk, J.[1,9]

1. Institute for Advanced Simulation (IAS-6), Jülich Research Centre, Jülich, Germany
2. RWTH Aachen University, Aachen, Germany
3. Department of Physics, University of Cagliari, Monserrato, Italy
4. Istituto Nazionale di Fisica Nucleare, Sezione di Cagliari, Monserrato, Italy
5. Istituto Nazionale di Fisica Nucleare, Sezione di Roma, Roma, Italy
6. Department of Psychiatry, Psychotherapy and Psychosomatics, School of Medicine, RWTH Aachen University, Aachen, Germany
7. Department of Physics, Faculty 1, RWTH Aachen University, Aachen, Germany
8. Department of Computer Science 3 - Software Engineering, RWTH Aachen University, Aachen, Germany
9. Sussex AI, School of Engineering and Informatics, University of Sussex, Brighton, United Kingdom

Summary and outlook

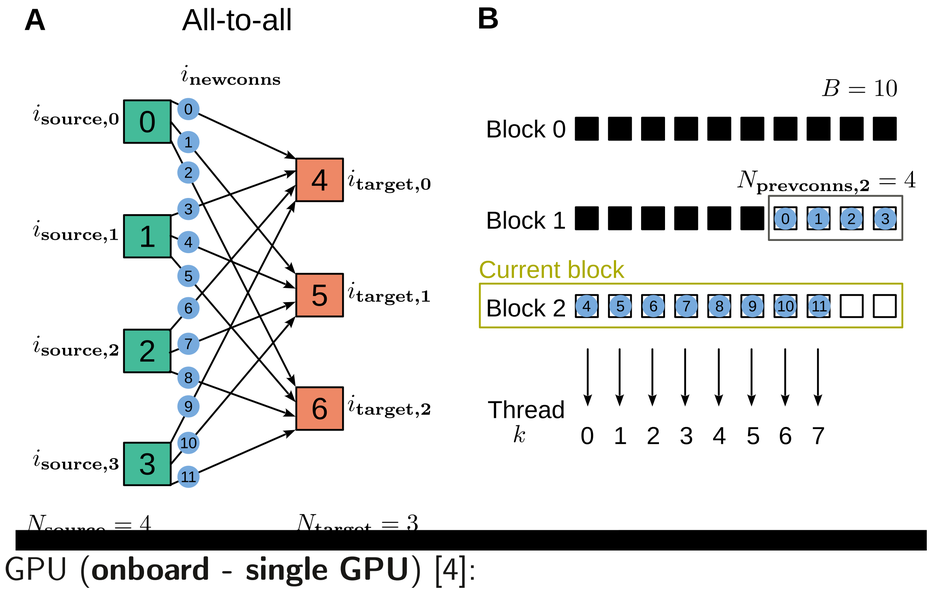
- **Fast network construction directly in GPU memory**
- **Tested on networks containing up to 21.6 million neurons and 243 billion synapses**
- **Future tests planned to evaluate performance impact with the multi-area model**

NEST GPU as part of the NEST Initiative

- Previously developed as NeuronGPU [1]
- Ongoing alignment to the community-centered workflow pioneered by NEST
- Multi-area model [2]: at least 2.4x faster simulation with NEST GPU compared to NEST, but 11x slower network construction time [3]
- NEST GPU (**offboard**): connections created between remote MPI processes are generated in CPU then moved to GPU memory

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

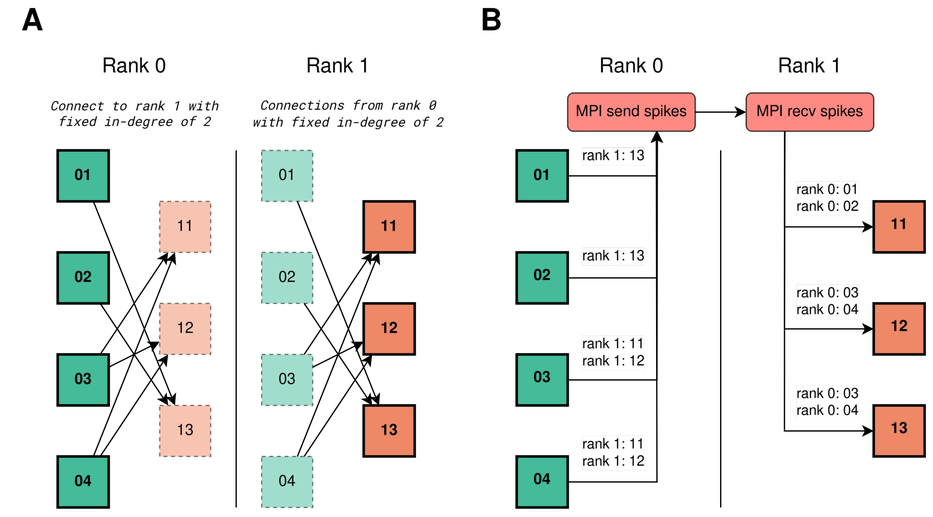
Dynamic allocation in GPU memory



NEST GPU (**onboard - single GPU**) [4]:

- Connections are **directly created in GPU memory** through dynamically allocated blocks with fixed number of connections per block
- Before simulation **connections are organized and grouped by delay** to optimize spike transmission (calibration phase)

Parallel connectivity generation in GPU

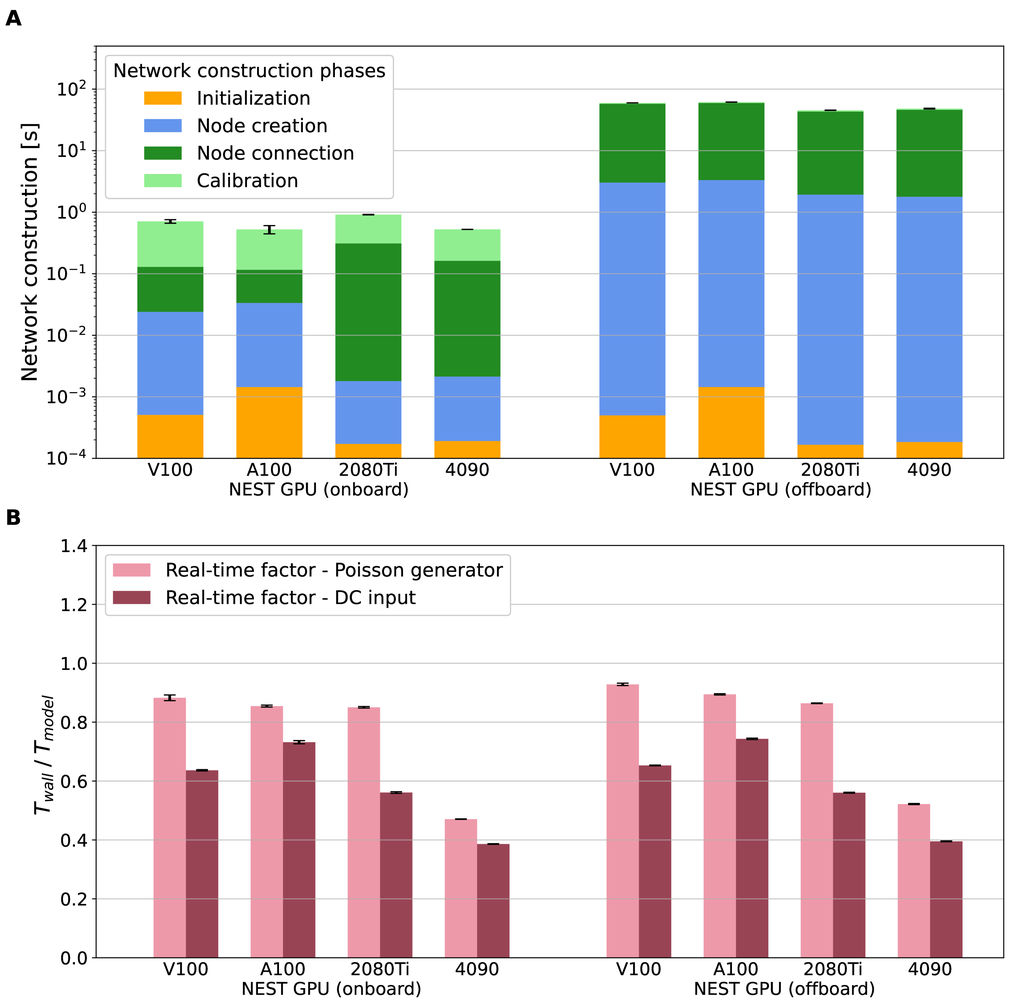


NEST GPU (**onboard - multi GPU**):

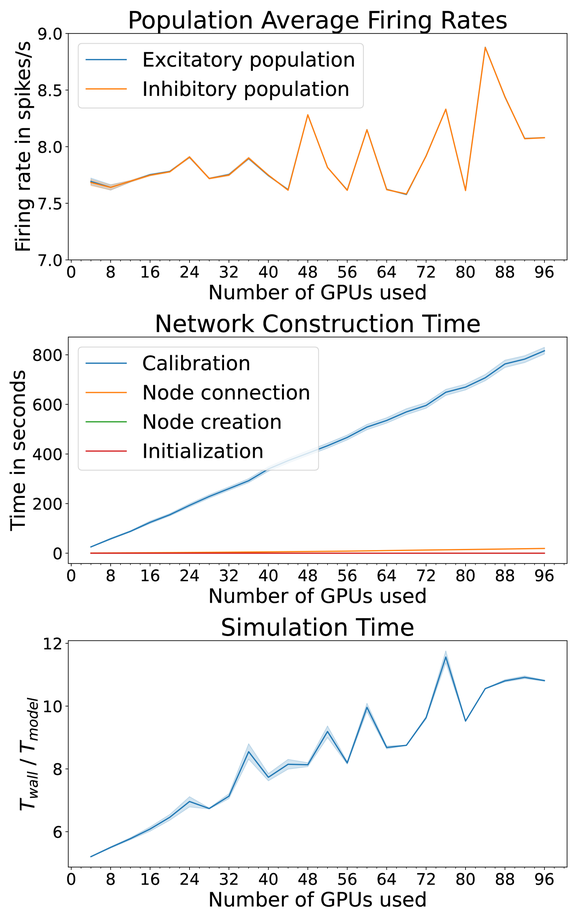
- **Mirrored use of pseudo-random number generators** ensure identical sequences of random states between MPI processes
- By synchronizing connection calls between MPI processes, **each process indepentently generates identical network structures**

Performance improvements on one GPU

Offboard and onboard method results using the **cortical microcircuit model** [5]:



Fast and scalable network construction



- Using the HPC benchmark, a **weak scaling model based on two interconnected neuron populations with fixed in-degree per neuron**
- Network parameters are defined so as to keep activity stable when scaling the total number of neurons
- Simulations ran in **JURECA DC-GPU partition with 4 NVIDIA A100 GPUs per node**
- Each GPU houses 225 000 neurons and 2.5 billion synapses
- **Largest network size is 21.6 million neurons and 243 billion synapses** (limited by maximum allocatable nodes for a single simulation run)

References

1. Golosio et al. "Fast Simulations of Highly-Connected Spiking Cortical Models Using GPUs", Frontiers in Computational Neuroscience 2021, 15. DOI: 10.3389/fncom.2021.627620
2. Schmidt et al. "A multi-scale layer-resolved spiking network model of resting-state dynamics in macaque visual cortical areas", PLOS Computational Biology 2018, 14, e1006359. DOI: 10.1371/journal.pcbi.1006359
3. Tiddia et al. "Fast Simulation of a Multi-Area Spiking Network Model of Macaque Cortex on an MPI-GPU Cluster", Frontiers in Neuroinformatics 2022, 16. DOI: 10.3389/fninf.2022.883333
4. Golosio, B. et al. "Runtime Construction of Large-Scale Spiking Neuronal Network Models on GPU Devices", Appl. Sci. 2023, 13, 9598. https://doi.org/10.3390/app13179598
5. Potjans & Diesmann "The Cell-Type Specific Cortical Microcircuit: Relating Structure and Activity in a Full-Scale Spiking Network Model", Cerebral Cortex 2014, 24, 785–806. DOI: 10.1093/cercor/bhs358

Acknowledgements

This project has received funding from Italian PNRR MUR project CUPI53C22001400006-FAIR, funded by NextGenerationEU; and Helmholtz Metadata Collaboration (HMC) ZT-I-PF-3-026. The authors gratefully acknowledge the VSR computation grant JINB33, Jülich, and the Fenix (ICEI project, grant agreement No. 800858). The authors acknowledge support from HiRSE_PS, the Helmholtz Platform for Research Software Engineering - Preparatory Study, an innovation pool project of the Helmholtz Association. The authors further thank the INFN APE Parallel/Distributed Computing laboratory and the IAS-6.