Human Brain Project

15-18 OCTOBER 2018 MAASTRICHT

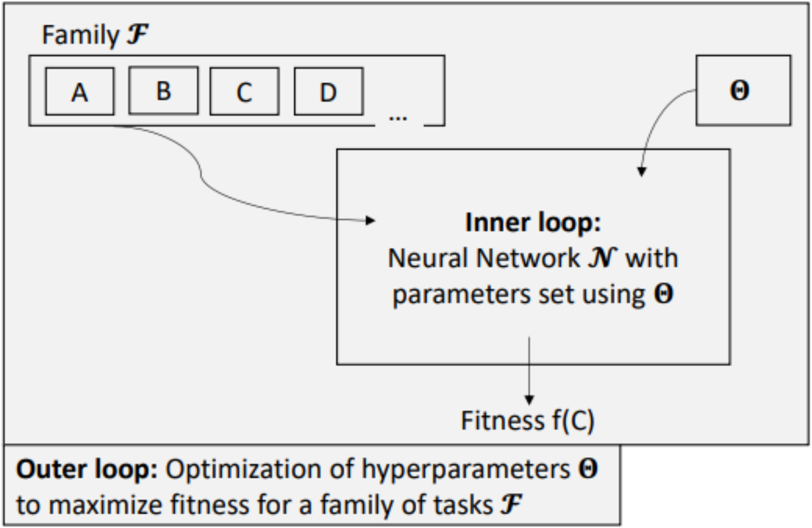
HUMAN BRAIN PROJECT SUMMIT

“Learning – 2 – Learn” on HPC

Alexander Peyser
Sandra Diaz & Wouter Klijn

SimLab Neuroscience
Forschungszentrum Jülich

L2L Structure



Bellec et al. 2018

Why use L2L on HPC?

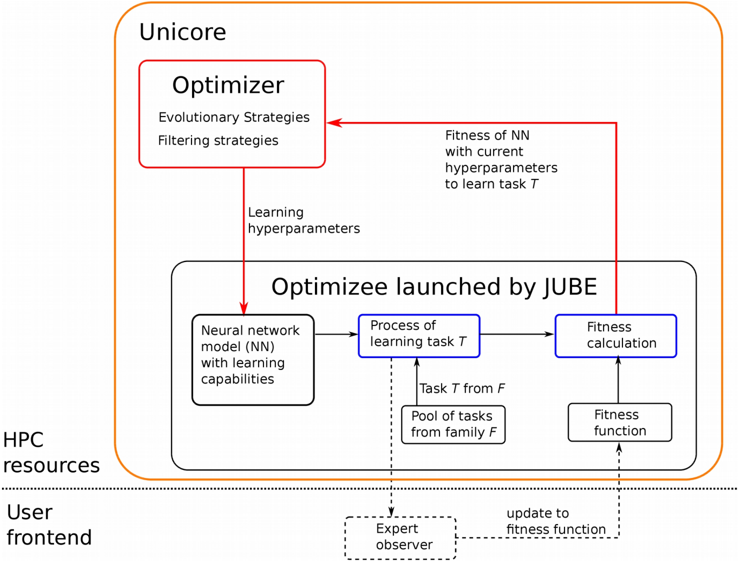
- To execute **computationally expensive** training runs.

  Optimizee needs significant resources to run in reasonable time, e.g. biological networks like the multi-area model
- To perform **extensive (hyper-) parameter optimizations** over large spaces

  Optimizee is light weight but many parallel instances need to be explored at one time like in parameter sweeps for TVB
- To couple external optimizees to **optimizers running on large data sets or requiring expensive computations** on HPC

  Optimizing neuromorphic networks and neurons
  Optimizing robotics applications

L2L architecture on HPC



L2L and parameter search flexibility

- The optimizee can have any form

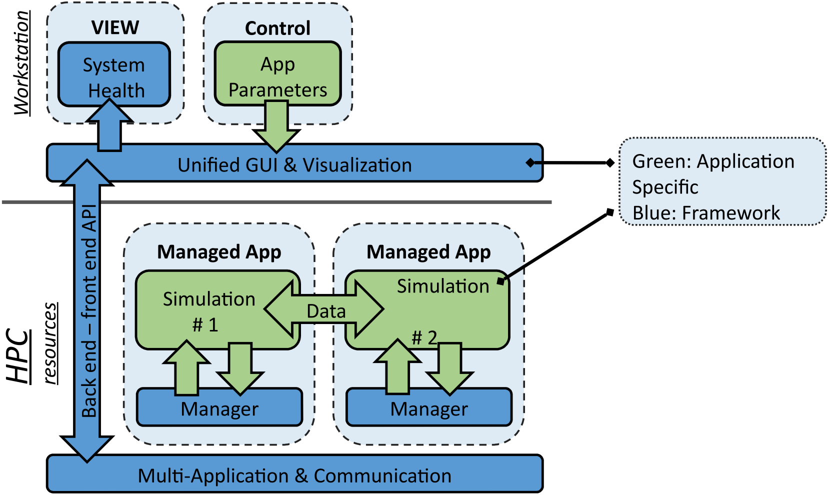
  Provides significant flexibility to the user

  A common set of meta-learning/search/optimization algorithms are in early development

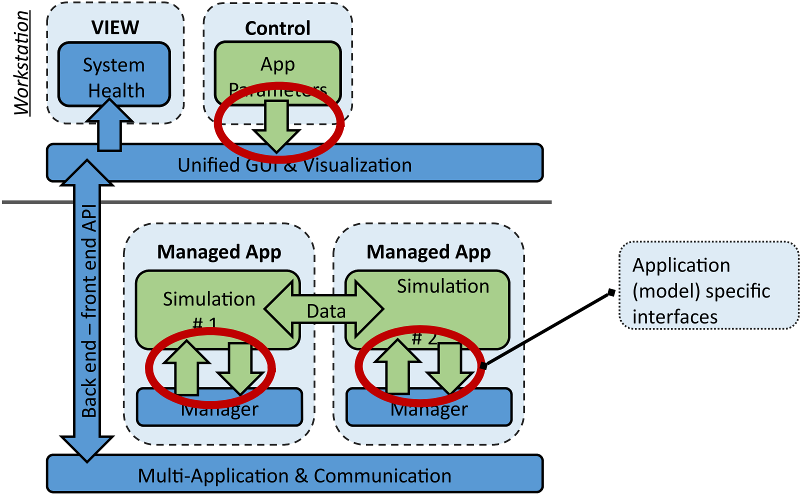
- Applicable to most scientific domains

  Driven by and specialized for neuroscience

L2L architecture on HPC: ModSci Framework

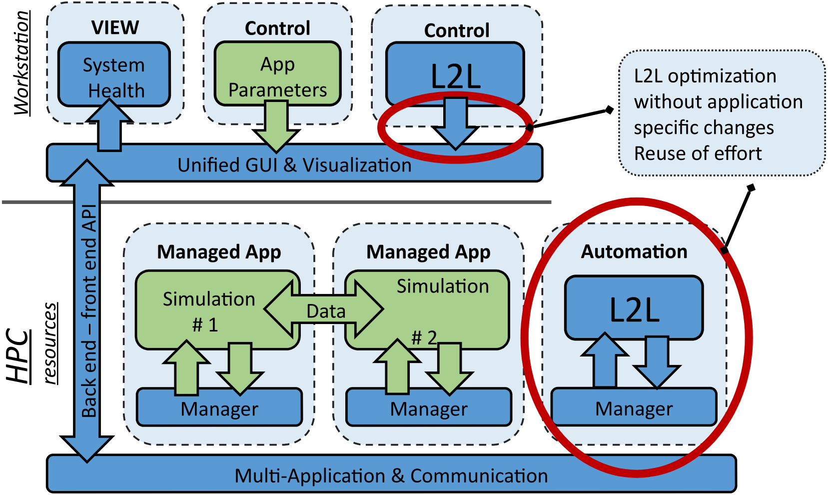


L2L architecture on HPC: ModSci Framework

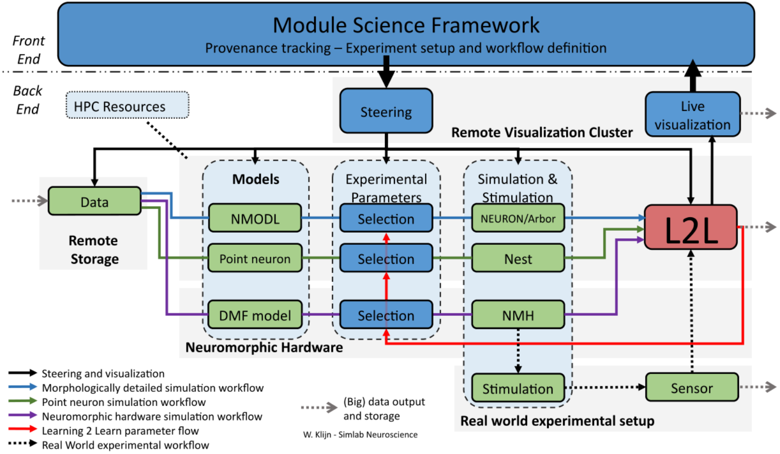


Wouter Klijn

L2L architecture on HPC: ModSci Framework



L2L architecture on HPC

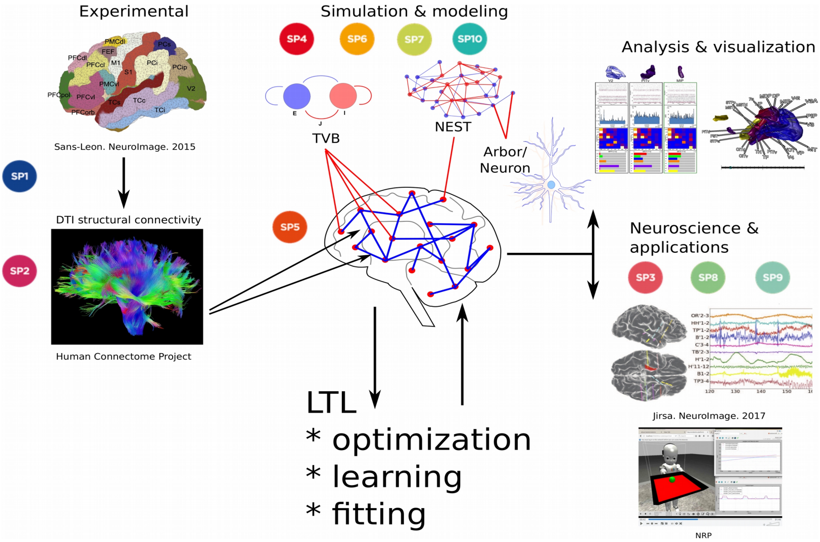


Wouter Klijn

L2L neuroscience workflow possibilities

- Parameter fitting/optimization of learning BNN models
- Exploration of hyperparameters
- Mixing standard fitness rules with expert knowledge scientist in the loop
- Online visualization of the progress of complex optimizees
- Visualization of the outer learning process

L2L neuroscience workflow possibilities: across domains



Sandra Diaz Pier

L2L: Infrastructure components

- **JuPeX**: L2L is currently implemented on HPC with JUBE plus on-going work with UNICORE
- **JUBE** is a benchmarking tool developed in Jlich
    - Previously: execution of functions/models using a brute force approach over a set of parameters
    - With L2L: parameters to be explored by JUBE are set by the outer loop optimizer
- **UNICORE** is a framework for the deployment of workflows on HPC
    - Allows very long optimization runs (from hours to months) in progress
- **ICEI**/Fenix infrastructure to support distributed storage
    - Future extension to improve support for external coupling of neuromorphic and robotic applications as well as elastic resources for time/generation varying demands

Thank you!

W. Maass
R. A. Legenstein
G. Bellec
A. Subramoney
T. Bohnstingl
A. Rao
D. Salaj
F. Scherr
J. Bennett

J. Knight
T. Nowotny
C. Pehle
S. Schmitt
E. Mueller
K. Meier
J. Schemmel
J. Jordan
M. Schmidt

M. Petrovici
W. Senn
S. Diaz- Pier
W. Klijn
A. Morrison
A. van Meegen
A. Korcsak-Gorzo
M. Diesmann
S. van Albada

Use case for Joint Platform

Training large ensembles of neuronal network on the machine learning platform

A scientist develops a simulation of a high-dimensional problem: for example, she wishes to train a spiking network with structural plasticity to quickly learn a generalized pattern from a family of exemplars (simulate the evolution of brains) or would like to identify stable fixed points for a neuronal network or neural mass model along a number of hyperparameters ("learning in the brain"), or train a network to drive an NRP robot in a variety of environments. She needs to run a large number of instantiations within the "Learning 2 Learn" (L2L, a branch of machine learning and AI) framework using an appropriate parameter search algorithm on HPC equipment and interactively identify hyperparameter regions of interest in order to efficiently use the resources available.

She works with the HLST to implement her simulation within the HPC Large-scale Workflow framework to allow her to independently run and customize the HPC instance of the simulations she's developed on workstations and small clusters. She chooses a set of available L2L search optimizations from algorithms developed by computer scientists, identifies measures of fitness, and selects a compatible visualizer. She then writes Python code (high-level programming language with low barrier of entry available on web portals) and appropriate configurations to begin HPC batch jobs, possible from a web-based portal. She connects the visualizer with her running jobs to monitor and interactively alter the long-running job as it develops. After many runs, she produces network configurations that have been evolved toward satisfying the constraints of the underlying problem. These are available on the virtual file system, and can be shared via the website with other researchers.

References