Technical Overview

Neronet

Toolbox for managing the training neural networks

CSE-C2610
Software Project

Aalto University

November 4, 2015

Aalto University
School of Science and Technology

Outline

Outline

Introduction

Neronet is a framework designed to facilitate the specification, submission, monitoring, control, analysis and management of many different computational experiments. The Neronet family consists of three members:

1. Neroman – a user run tool that acts as the *Neronet frontend* and user interface to provide access to Neronet's features and functionality. It is the researchers workhorse.
2. Neromum – a Nerokid manager daemon that helps the family stay organized while the kids are playing in distant clusters. Neromum is designed to collaborate with a `warden` (f. ex. SLURM, OGE or Jobman) if necessary
3. Nerokid – a tiny creature that looks after your computational toy and reports to mum in case it breaks.

Introduction

In more technical terms:

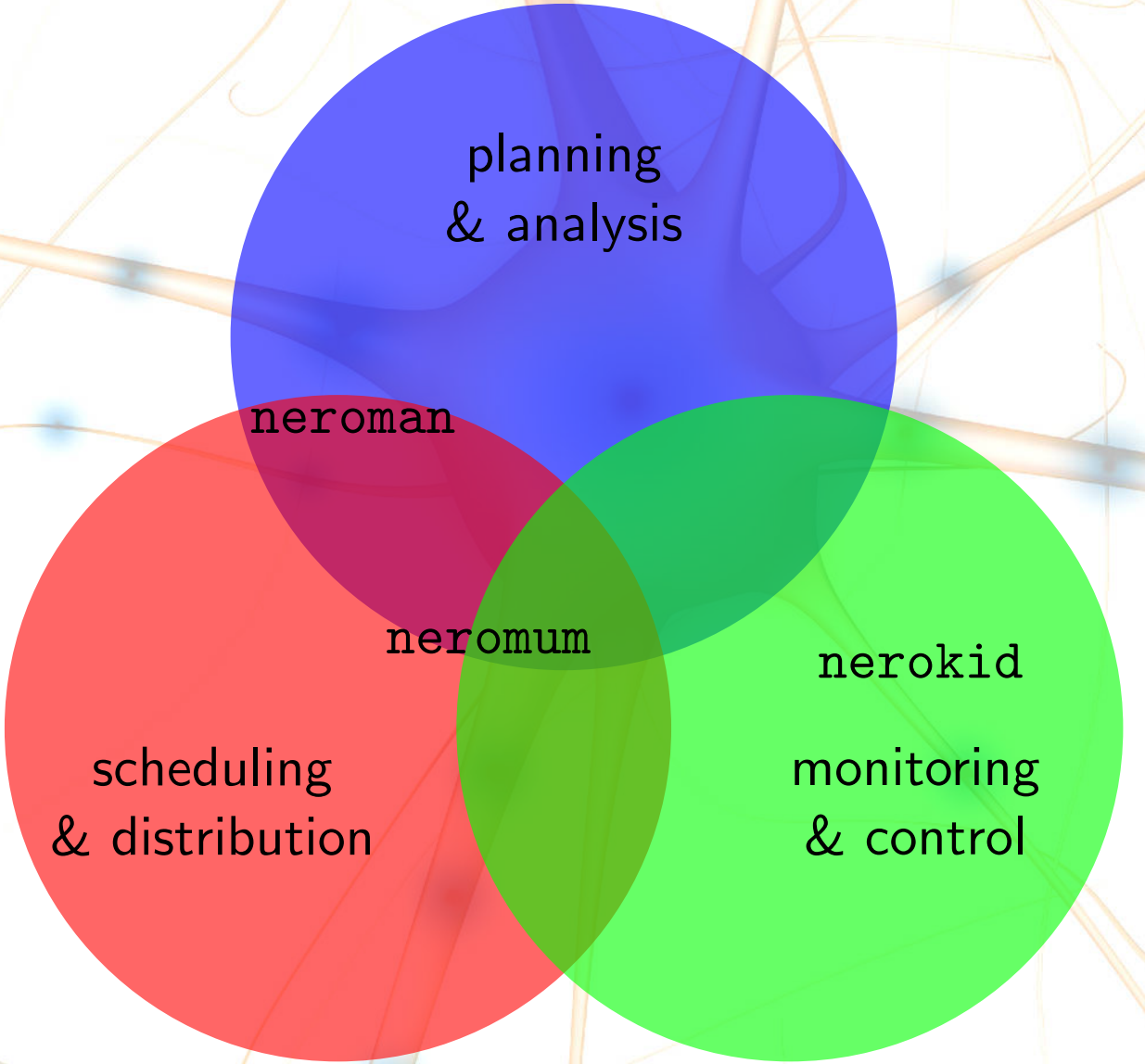
- Nerokid is a program designed to be launched by a job scheduler or Neromum directly in a cluster node to monitor, analyse and control a single experiment job and its environment in tandem with a Neromum
- Neromum is a job scheduler, manager and communications agent for Linux nodes or clusters designed to work by herself or with an existing `warden` (job scheduler)
- Neroman is a tool that acts as the *Neronet frontend* and user interface to provide Neronet's functionality by dispatching Neromums in specified clusters or nodes with a bunch of jobs to do. The Neromums in turn dispatch the jobs to the Nerokids and then continue as communication intermediaries.

Introduction

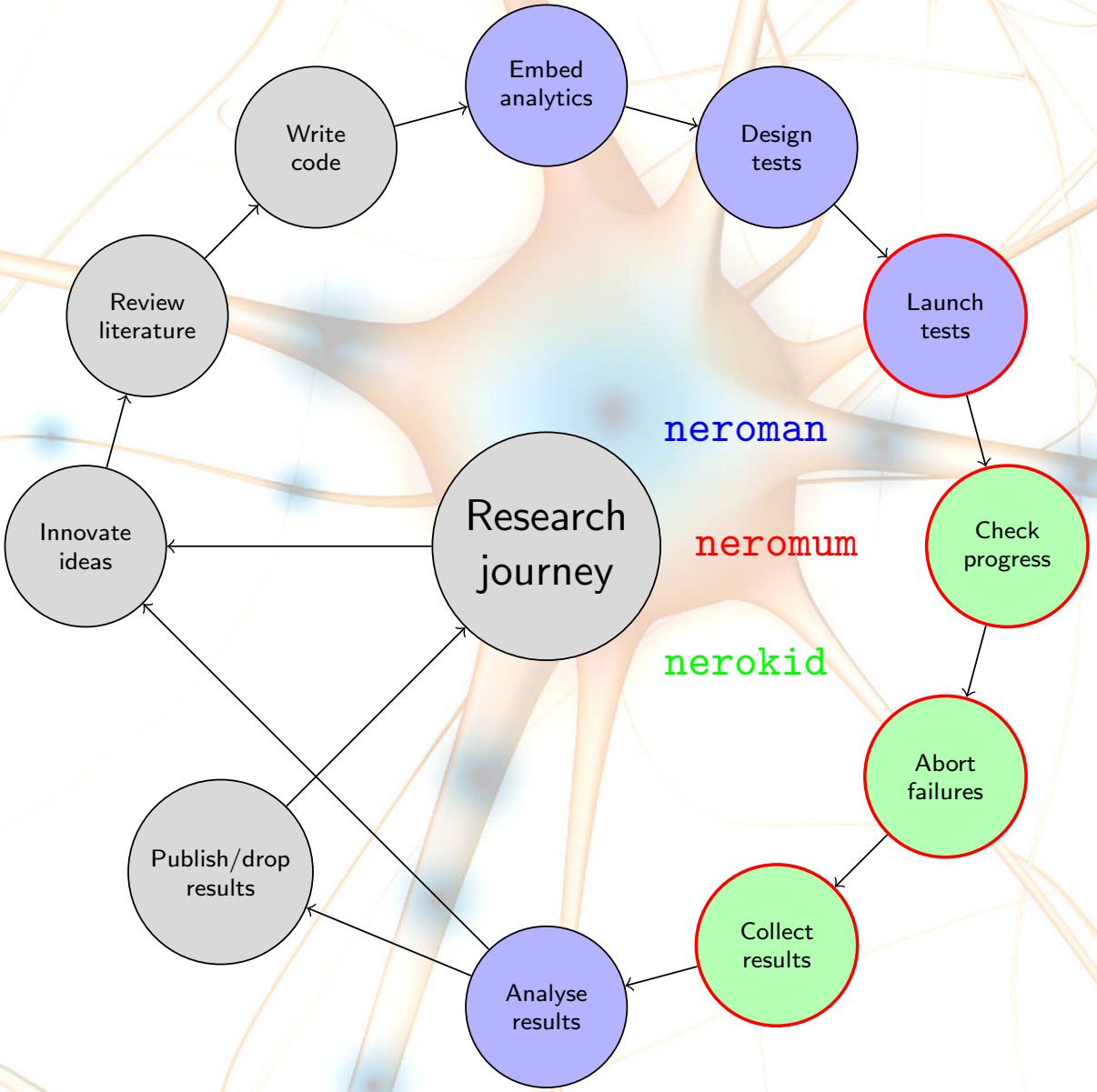
The following three slides provide a good introduction:

1. *Job management* – the associated problem domain spheres and how the system components are related to them
2. *Journey map* – the journey or work flow of a typical system user (a computational researcher) and the steps in which the Neronet tools are designed to be used
3. *Basic use case* – A sample basic system use case description

Job management



Journey map


Embed analytics
Write code
Design tests
Review literature
Launch tests
neroman
Research journey
neromum
Innovate ideas
Check progress
nerokid
Abort failures
Publish/drop results
Collect results
Analyse results

Basic use case

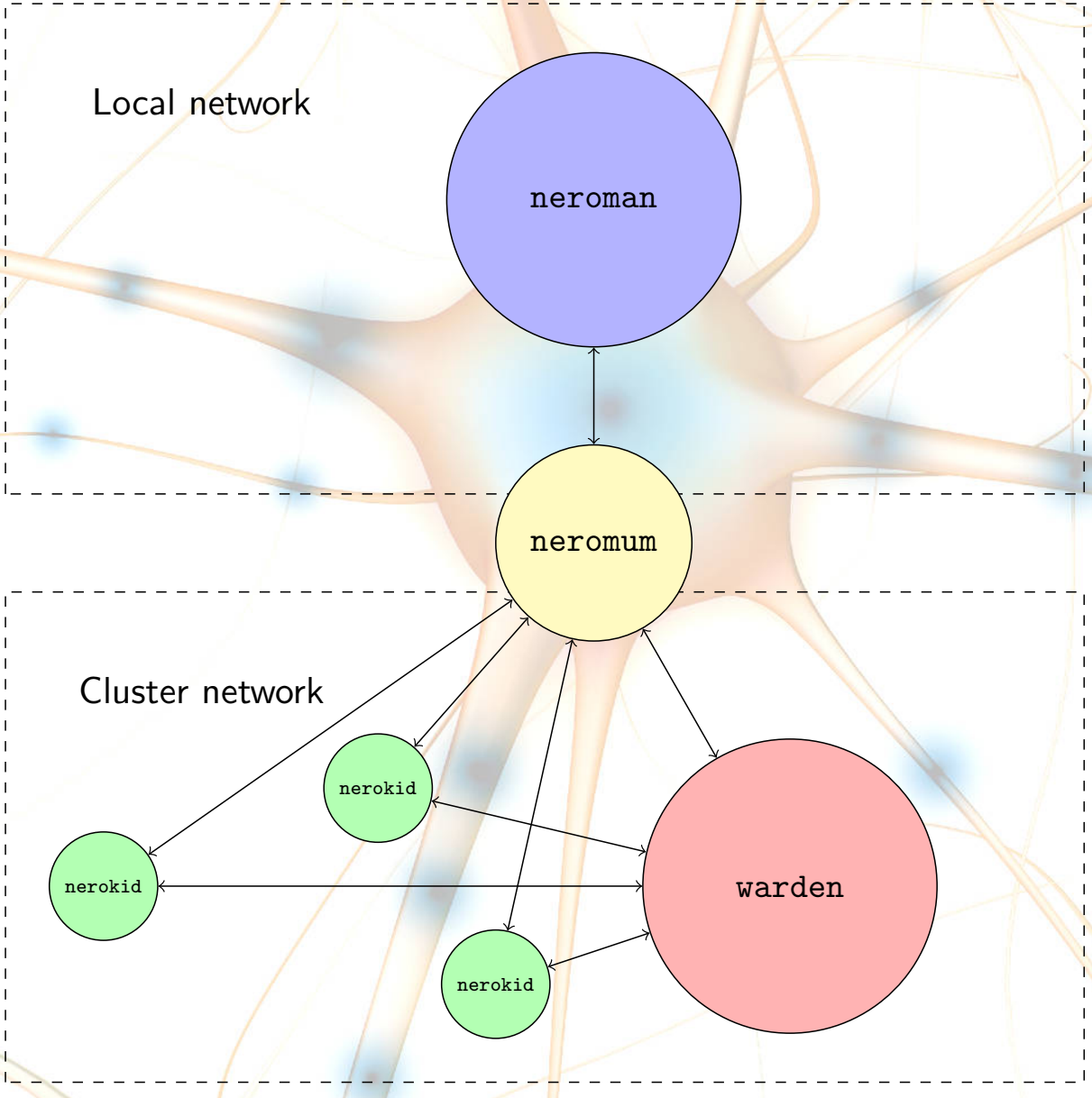
- **User**: A computational researcher
- **Goal**: To test how well a new design works with several different configuration options and parameter values
- **Preconditions**: SLURM cluster and Neronet setup, code and analytics developed and test inputs setup in Neronet compatible manner
- **Basic flow**:
  1. Specify a batch of Nerokids in the *experiments excel* with parameters, inputs and other configurations
  2. Dispatch the jobs to the SLURM setup with autogenerated `sbatch` scripts and arguments: `neroman --start 124-186` or `neroman --start 'tmod<1h'`
  3. Receive and check progress notifications from email
  4. Reconfigure some jobs in excel and restart them: `neroman --restart 'crashed,144-149,170,175'`
  5. Abort some others: `neroman --abort 'runtime>7200&&errorrate>0.40'`
  6. See near realtime updates in the *results excel* of analytics variable updates
  7. Receive final results data and updates to the job excel as configured
  8. Analyse, reiterate and/or publish results
- **Post conditions**: Computational experiments have been conducted in a very straightforward, effective and researcher friendly fashion

Outline

Components

- All Nero components are lightweight Python programs run with just the researcher's privileges on any modern *nix
- SSH is used for communication between Neroman and Neromums (user's existing ssh keys, ssh configs and privileges are used)
- Sockets are used between Neromums and Nerokids
- Neromum communicates with any `wardens` using their CLIs and/or APIs
- The system is ment to be easy to setup, lightweight and the usability good for several types of uses.

Communication



neroman

- A daemon administered and configured by the researcher herself
- Should be run on a system with two way SSH access to the cluster gateway or nodes where the Neromums are deployed
- Key functionality
  - Facilitate and standardize experiment specification
  - Batch submit experiment jobs to a `warden` with autogenerated job scripts (eg. `sbatch`)
  - Send email notifications with progress data
  - Facilitate monitoring and control of running jobs
  - Autocollect key job results into a researcher specifiable format (f. ex. Excel)
  - Facilitate experiment analysis and history management
  - Lightweight and extendable with custom functionality
  - Configurable via YAML files

neromum

- A daemon started and administered by a Neroman.
- Should be run on a system with two way SSH access to the Neroman and socket access to the nodes where Nerokids are deployed
- Ment to work with `wardens` – standard job scheduling systems for Linux clusters such as SLURM, OGE or Jobman.
  - Each has both CLI and API interfaces and at least one of them is expected to be presetup by research systems administrators.
  - SLURM is currently used by Triton and CSC. OGE has been used a lot in the past worldwide. Jobman could be easily setup for the gpu cluster used by deep learning researchers at Aalto.
- Key functionality
  - Collect information from Nerokids and send them to Neroman
  - Transmit commands from Neroman to the Nerokids

nerokid

- A daemon started and administered by a Neromum on any modern *nix cluster node to start and monitor computational jobs
- Key functionality
  - Fetch and send computing environment information to Neromum
  - Monitor experiment job progress (parse output logs and data files (eg. CSV, JSON))
  - Autocollect and send information and data to Neromum
  - Interact with the `warden` as specified (eg. autotermination based on poor experiment progress)
  - Lightweight and extendable with custom functionality
  - Configurable via YAML files

Outline

Remarks

- A server (`neroman` daemon) per user approach is chosen because
  - easy minimalist setup (easy to try)
  - no need for special privileges
  - fully customizable by the user herself
- SSH is used because
  - it is an existing standard in most modern research systems and clusters
  - user's existing ssh keys and configs provide an easy and effective way to provide safe networking
  - no need for network, port routing or privileges adjustments
- Python 3 is used because
  - available in most modern *nix systems
  - has good support for the system's needs
  - it is easy to learn and familiar to most computational researchers
  - Many research libraries use it (Scipy, Numpy, Theano, Lasange, Pylearn, Blocks)