User guide

CLI

Demo

```
neroman --cluster mycluster localhost none
neroman --experiment ~/experiments/sleep
neroman --submit mycluster sleep2
neroman --monitor sleep2
```

Defining cluster setup

Using the following command Neronet defines a new cluster into its database and tests whether it can access and operate with it.

```
Usage: neroman --cluster ID SSH_ADDRESS TYPE
Example: neroman --cluster triton triton.cs.hut.fi slurm
```

Defining experiments

```
Usage: neroman --experiment FOLDER
Example: neroman --experiment ~/experiments/lang_exp
```

Experiment folders must include a YAML config file named `neronet.yaml` of the following format:

```
# EXPERIMENT_ID: RUN_COMMAND PARAMETERS
lang_exp1: python3 lang_exp.py 1 2 3 data/1.txt
lang_exp2: python3 lang_exp.py 2 2 1 data/1.txt
lang_exp3: python3 lang_exp.py 3 2 1 data/2.txt
```

The experiment IDs must be unique.

Submitting experiments to be run

```
Usage: neroman --submit CLUSTER_ID EXPERIMENT_ID
Example: neroman --submit triton lang_exp3
```

Tasks can be submitted also by logical arguments:

```
Usage: neroman --submit CLUSTER_ID ARGUMENT
Example: neroman --submit triton ~/experiments/lang_exp
Example: neroman --submit triton 'tmod>yesterday'
Example: neroman --submit triton 'params=*data/1.txt*
```

Checking status

The status command gives status information regarding configurations and any specified clusters and experiments.

```
Usage: neroman --status [ARGS]
```

ARGS can refer to experiment or cluster IDs, or be collection specifiers.

```
Example: neroman --status # Overall status information
Example: neroman --status lang_exp3 # experiment status
Example: neroman --status 'tsub>yesterday' # collection status
Example: neroman --status triton # cluster status
```

Monitor experiment progress

```
Usage: neroman --monitor EXPERIMENT_ID
Example: neroman --monitor lang_exp3
```

GUI