Running NetLogo BehaviorSpace Experiments on the Cluster.

This is a hands-on guide. It does not aim to be fully comprehensive, but just to get yourself started with running NetLogo BehaviorSpace experiments on the cluster. For running experiments in general on the cluster there is an introductory presentation [here](https://docs.google.com/presentation/d/10A0_0eNRBYd87E1h1YN6bsIFaZaua5qJkfBbnBKAr6o/edit#slide=id.g598656af69_0_0). Further, the full [documentation](https://login.hpc.tudelft.nl/) is available on the TU Delft website.

This document is written by Vittorio Nespeca. It is inspired by and builds on a previous document written by Indushree Banerjee. The document is also based on suggestions by Martijn Warnier.

# Step 1: Request access

Request access to the cluster (if PhD/Postdoc ask your supervisor to do it on your behalf).

# Step 2: Running experiments locally via the command line

Before diving into using a remote machine via the command line, it’s best to get yourself acquainted with running netlogo experiments on your local machine though the command line.

* Design your experiments and save them in the NetLogo GUI (they are saved at the end of your NetLogo model file, so they come with it). You can also write the experiments separately as XML files (we will not do that here).
* Run some behavior space experiments on the NetLogo GUI to get yourself familiar with it and see if there are any bugs to be fixed.
* Learn how to run experiments with NetLogo via the command line.

All of the above is explained in the NetLogo documentation on BehaviorSpace (latest link at the time of writing [here](https://ccl.northwestern.edu/netlogo/6.0.2/docs/behaviorspace.html)).

# Step 3: Setting up and managing your remote storage

Here we set up our remote storage by transferring all the files and installing the software required to run our simulations.

## Learn how to login in two steps:

1. Login to bastion:
   1. In your local terminal type:

*ssh net-id@linux-bastion.tudelft.nl*

* 1. Input your password and press enter
  2. You are now logged in in the bastion

1. From bastion, login to cluster.
   1. Type the following

*ssh login.hpc*

* 1. Insert your password and press enter.
  2. You are now logged in into the cluster.

The cluster is the place where you can run your model (not the bastion). Yes, you have to type your password twice (I know, not the most practical…). There is a way explained [here](https://wiki.tudelft.nl/bin/view/Research/InsyCluster/ConnectingSSH) to set it up so that the extra step is not needed (check the section “Making OpenSSH more user-friendly”).

## Fundamentals of Bash: handling files and folders from the command line

It’s no big deal. But you’ll need to be familiar with removing, moving, copying, accessing, and modifying folders and files in bash. If you are not familiar with it. Or, if you need to refresh your memory, check out commands like ls, pwd, cd, rm, cp, cat and nano.

## Installing NetLogo on the remote machine

1. Download netlogo for linux. The zipped installation file is something like NetLogo-6.1.1-64.tgz
2. Transfer the netlogo file to the remote machine with scp:
   1. From your local terminal/command line move to the location where you downloaded the netlogo file (use the command cd)
   2. Then, type:

*scp -p NetLogo-6.1.1-64.tgz net-id@linux-bastion.tudelft.nl:*

* 1. Input your TU Delft password and press enter
  2. Your netlogo installation file has been copied to the remote machine

1. Login into the cluster (above)
2. Type *ls* in the command line and press enter, you will see that *NetLogo-6.1.1-64.tgz* is there.
3. unzip the file using the command tar zxvf. Type the following:

*tar zxvf NetLogo-6.1.1-64.tgz*

1. The file will be “installed” in your local machine
2. We also create a soft link to netlogo using *ln -s*

*ln -s NetLogo\ 6.1.1/ netlogo*

1. You can now access the netlogo folder by typing *netlogo* instead of *NetLogo\ 6.1.1*

## Uploading Netlogo model and other necessary data

1. Log out from the cluster and bastion (type *logout* and enter twice)
2. From your local terminal/command line upload your netlogo model and all the files it requires as a zipped folder using *scp* (as shown above for the netlogo installation file)
3. Unzip the files in a convenient location, we’ll need it when setting up our jobscript.

# Step 4: Setting up & running your cluster experiment

Here we are setting up a virtual machine with the “hardware” that we require to run our experiments. Think of it as choosing your new computer based on what you need. We will have to set the resources required as in the number of CPUs and RAM. We will also have to set the quality of service and time required to perform our experiments.

## Fundamentals of Slurm

Slurm is the software used at TU Delft to manage the cluster. It allows you to set up your “new computer” based on resources available in the cluster and run experiments in batches within it. The language spoken by this software is the following:

* Job: is the batch of tasks that you can run on the cluster
* Tasks: 1 task correspond to one program you are running (e.g., our netlogo program, or a python script). One job can run multiple tasks/programs in parallel.
* cpus-per-task: number of cores that are assigned to each task (i.e., each program). This point is quite crucial as netlogo automatically finds how many CPUs are assigned to the program (task) and runs as many experiments in parallel.

Key commands for Slurm are:

* Running a job:
  + *sbatch*: runs a job, summarized in a jobscript (shown later)
  + *srun*: runs a task from within a jobscript
* Checking and managing a job that is running or was run in the past:
  + *squeue --user=net-id* : shows the jobs you are currently running and their job-ids
  + *scancel job-id* : cancels a job
  + *scontrol show job job-id* : shows detailed information about the job
  + *sacct -j job-id* : shows the current status of the job
  + *sacct -l -j job-id* : shows very detailed info on current/past jobs, including how long it took to run a completed job. It looks messy.
  + *cat slurm-job-id-out*: shows the file that contains all the messages incoming from the program(s) run in the batch. This file is your main interface with the batch you are currently running or run in the past and it includes any errors and warnings encountered. Also, anything you “echo” in your jobscript will end up in this file.

## Writing and running your first job script

The job script is a text file with the extension “.sbatch”. An example script for running experiments in the cluster is shown below. We’ll call this script *“happy-jobscript.sbatch”*. For a thorough explanation of the –qos command (Quality of Service) please check the presentation [here](https://docs.google.com/presentation/d/10A0_0eNRBYd87E1h1YN6bsIFaZaua5qJkfBbnBKAr6o/edit#slide=id.p).

#!/bin/sh

# The default partition is the 'general' partition

**#SBATCH --partition=general**

# The default Quality of Service is the 'short' QoS (maximum run time: 4 hours)

**#SBATCH --qos=short**

# The default run (wall-clock) time is 1 minute. In this case it is set to one hour.

**#SBATCH --time=1:00:00**

# The default number of parallel tasks per job is 1. We set it to 1 as we are running 1 netlogo task

**#SBATCH --ntasks=1**

# Request 1 CPU per active thread of your program (assume 1 unless you specifically set this)

# The default number of CPUs per task is 1 (note: CPUs are always allocated per 2)

# This number defines how many experiments in parallel the netlogo task will carry out

**#SBATCH --cpus-per-task=14**

# The default memory per node is 1024 megabytes (1GB) (this is the RAM of our “new computer”)

**#SBATCH --mem=4096**

# Set mail type to 'END' to receive a mail when the job finishes (with usage statistics)

# Do not enable mails when submitting large numbers (>20) of jobs at once

**#SBATCH --mail-type=END**

# tell netlogo where to find java jdk, this depends on what versions are available and you choose.

# check what you have with cd ~/usr/lib/jvm/ and then pick one of the versions in the folder (use ls)

**export JAVA\_HOME="/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.302.b08-0.el7\_9.x86\_64"**

# run the netlogo program as you did for your local machine but this time on the remote machine

**srun ~/netlogo/netlogo-headless.sh --model experiments-and-data-analysis/share\_descriptive\_1.0\_hpc.nlogo --experiment sample-run-1 --table results-sample-run.csv**

**echo "Script finished"**

Code snippet: Content of the text file happy-jobscript.sbatch

Once the script is ready you can run it from the cluster command line using the command:

*sbatch happy-jobscript.sbatch*

## Figuring out the Resources required to run your NetLogo experiments.

How to understand how many CPUs, memory and time required to run the experiments? It’s a bit of an iterative process in which you start with small experiments with a short duration, and you work your way up to the actual experiments you need to run. You can use the instructions [here](https://docs.google.com/document/d/14-jokK4MLbnJk44QaAtIH2LnxrVBooA6u3ZqI6nrPpw/edit#heading=h.cnd98i4fxjmt) under the heading “How to determine the CPU load, memory use and number of active threads of your program?”. This will give you an initial idea, but you can actually use more cores and memory if the computational time is too long.

## Setting enough memory to avoid running into Out Of Memory errors

NetLogo is written in [Scala](https://www.scala-lang.org/) and uses a java virtual machine (JVM) to run the program (your model experiment) within the virtual machine that you set in the cluster (we called it our newly set computer). So, we have a virtual machine within another one (Inception vibes are real). If we don’t provide enough RAM memory to both our virtual machines, the job will be automatically interrupted.

### Providing enough memory to the JVM:

If we get an error of the type:

*§Exception in thread "JobThread" java.lang.OutOfMemoryError: GC overhead limit exceeded*

This means we haven’t provided enough memory to our JVM. To change this memory, we will need to access the netlogo-headeless.sh file which is placed in the *netlogo* installation folder. We’ll need to change the number highlighted in the following figure to the amount of memory needed (the number is in megabytes).

**A screenshot of a computer

Description automatically generated with medium confidence**

Changing the amount of memory available for the Java Virtual Machine set up to run NetLogo.

### Providing enough memory to the external virtual machine:

If we get an error of the type:

*slurmstepd: error: Detected 1 oom-kill event(s) in step \*\*\*\*\*\*.batch cgroup. Some of your processes may have been killed by the cgroup out-of-memory handler.*

This means we haven’t provided enough memory to the external virtual machine (our newly set computer). We can adjust that with the *--mem* command in your jobscript text file (see above). The memory of your computer must be higher than that of the JVM. I would presume that’s because the external virtual machine needs extra memory to run the internal one.

# Step 5: Resilient experimental setup

Surprise! Jobs in Slurm can fail. I know, it’s not the most comforting… A job may take hours, days, weeks, or longer! Setting up one big job and submitting it is risky. We could lose all the results and time invested. How can we be resilient about this? Setting up modular experiments can help. Here’s how to do it:

1. Break up your experiments in different chunks. Say that we want to explore the impact of 4 different parameters. Take the parameter for which you are exploring the highest number of different values and set up one separate behavior space experiment for each of the values. This will cut down the number of total runs for each experiment.
2. Run each of the experimental chunks as an individual and separate job on the cluster.
3. You can easily reassemble the results by adding up to the “[run number]” column in the csv file in which netlogo saves the results.

# Step6: Copying the results back to your local machine

Once your jobs are completed you will need to copy the files with the results back to your local machine. We can do that with scp as in the following

*scp -p net-id@linux-bastion.tudelft.nl:/path/toresultsfile/results-sample-run.csv*

*path/tolocalfolder/wheretosavefiles/results-sample-run.csv*

This should be done from the command line/terminal of your local machine.