Cloud Parallel Computing

Evaluating `R` code in parallel

Wael Mohammed

Table of contents

0.1 Introduction:

This document discusses creating, interacting with and using cloud computing.

0.2 Create an account in Linode:

- Go to https://login.linode.com/signup and register for a new account. At the time of writing, Linode was running a promotion where new users are given *$100* worth of credit to spend within the platform.

Warning: Remember to shut down and delete all Linode instances once done with them to avoid unintended costs.

- Log in to the Linode account at https://login.linode.com/login.

0.3 Creating a Linode instance:

From the Linode dashboard:

- Select **`Linodes`** from the left side menu.

- Click on **`Create Linode`**, and several tabs will populate the page.

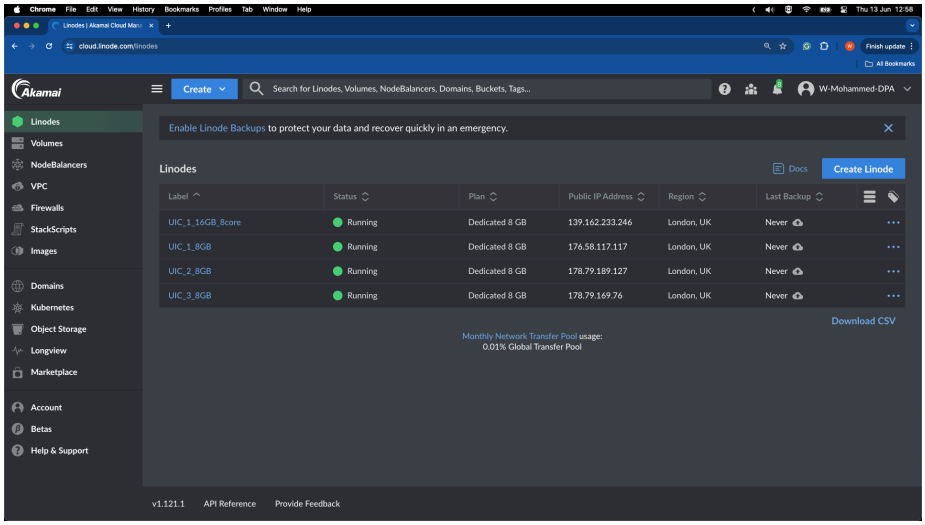

Figure 1: Linode Dashboard

0.3.1 Distributions:

- **Choose a Distribution**: Select a suitable operating system image.
  - Select `Ubuntu 24.04 LTS` from the list of images.

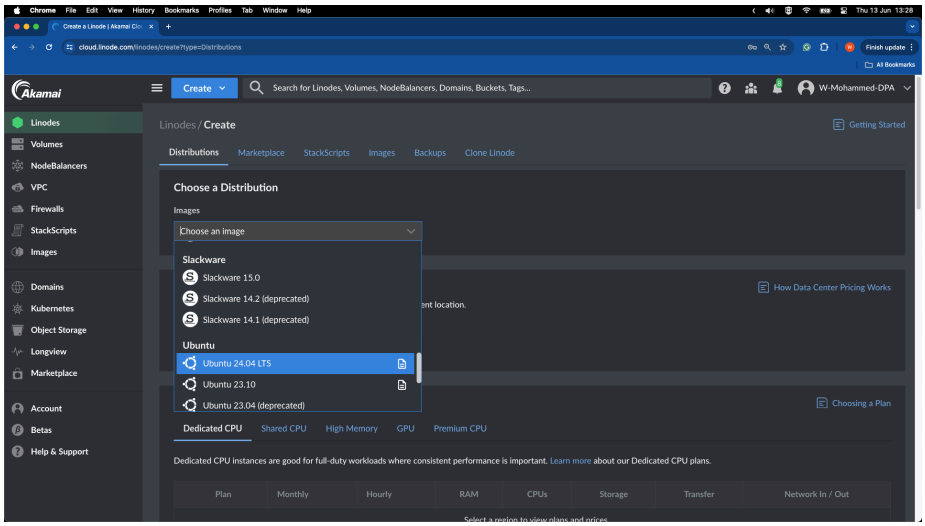

Figure 2: Selecting a Linode Operating System

- **Region**: Choose the closest location.

– Choose `London, UK (eu-west)` from the list of regions.

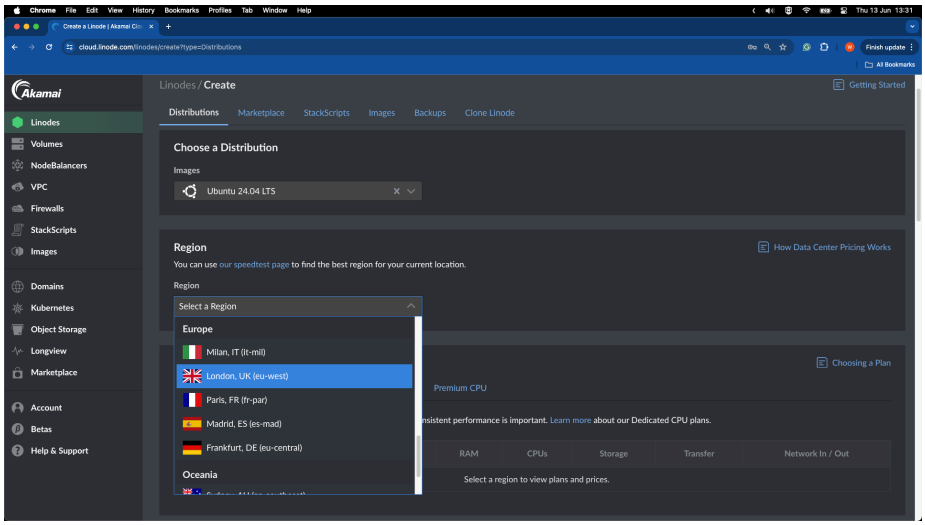

Figure 3: Choosing a Linode Region

- **Linode Plan**: Choose an appropriate computing node.
  – Choose `Dedicated 8 GB` from the list of plans.

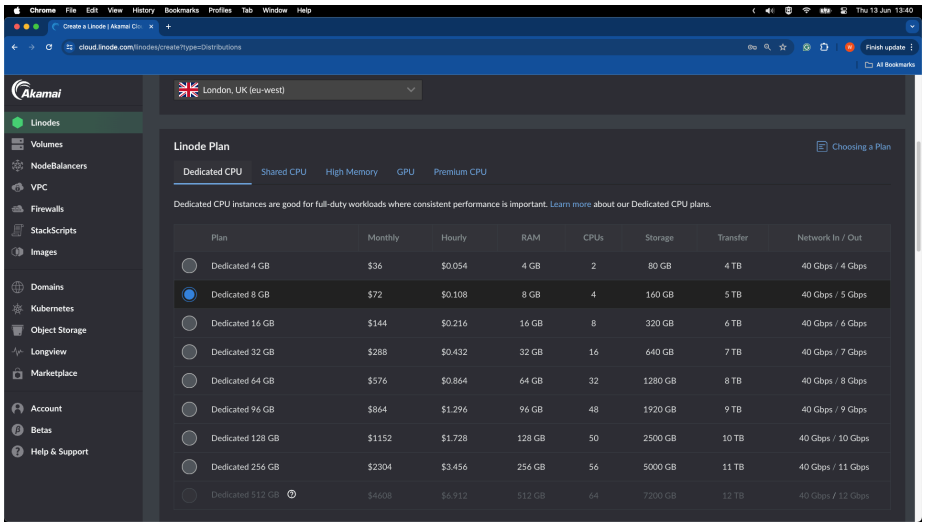

Figure 4: Choosing a Linode Plan

- Linode **Details**:
  – **Linode Label**: Give the cloud computing instance a meaningful label.

* Use `UIC_4_8GB`

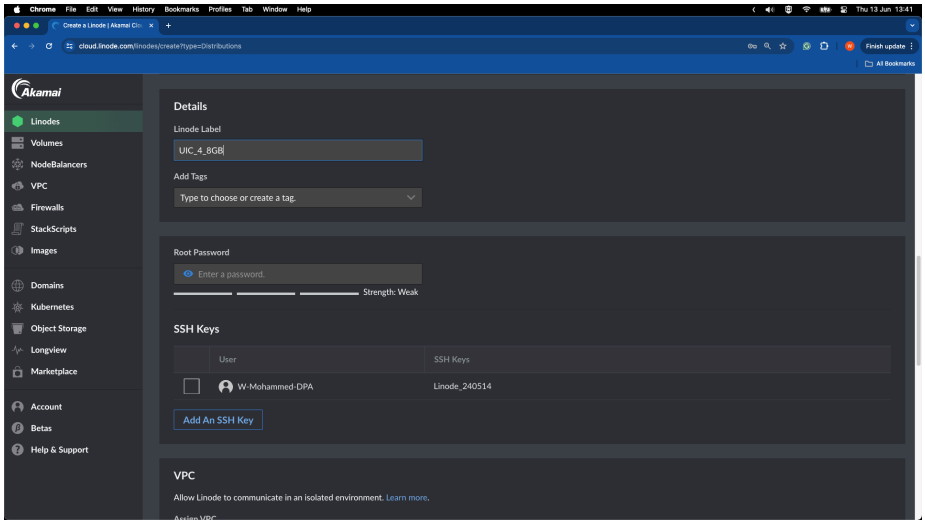

Figure 5: Labelling a Linode Instance

- **Add Tags**: A tag if required.
- **Root Password**: Set a root password.
- **SSH Keys**: Click `Add An SSH Key`, paste your `SSH public key` and label it. To learn more about using SSH and SSH keys, check:
  - https://www.linode.com/docs/guides/connect-to-server-over-ssh-on-windows/
  - https://www.youtube.com/watch?v=ZVMckBHd7WA

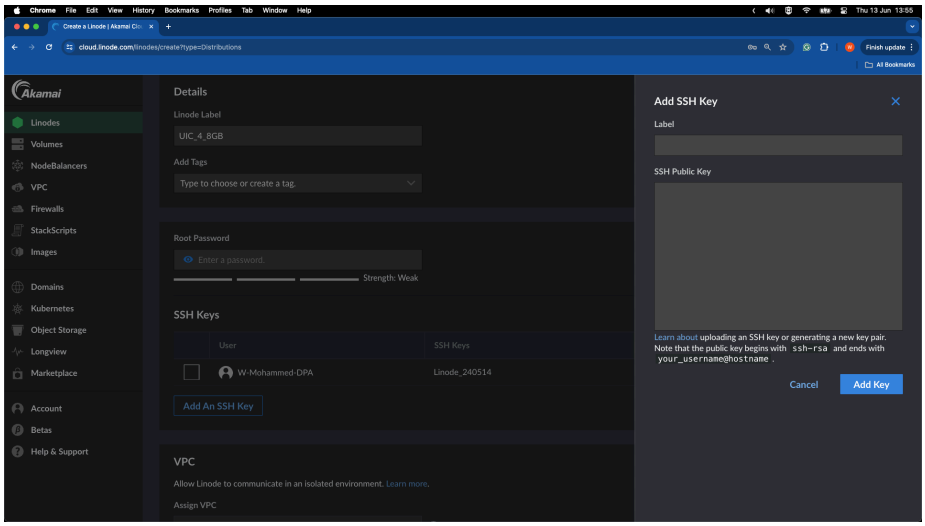

Figure 6: Using SSH Key

0.3.2 Initialise Linode Instance:

Click `Create Linode` to confirm the details and spin the chosen instance.

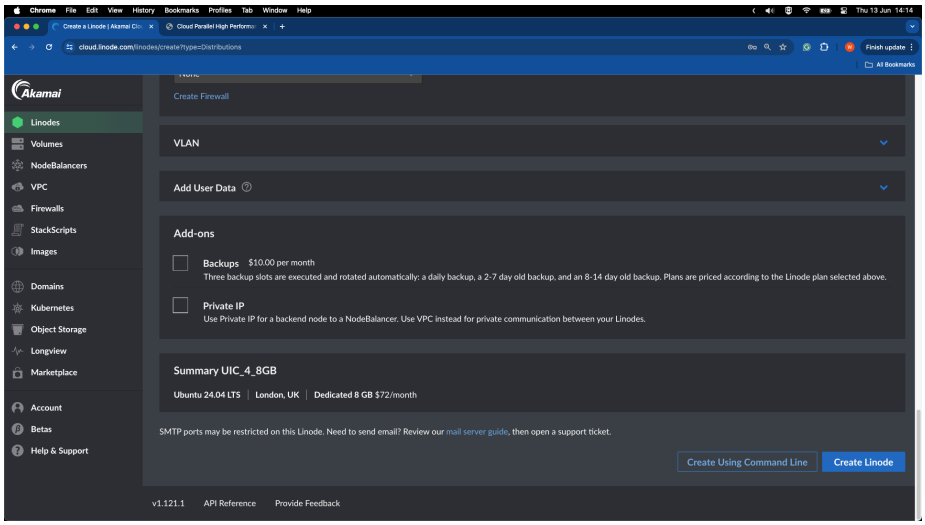

Figure 7: Create Linode

The Linode instance will start spinning, and the page will show "`Provisioning`", "`Booting`" and "`Running`".

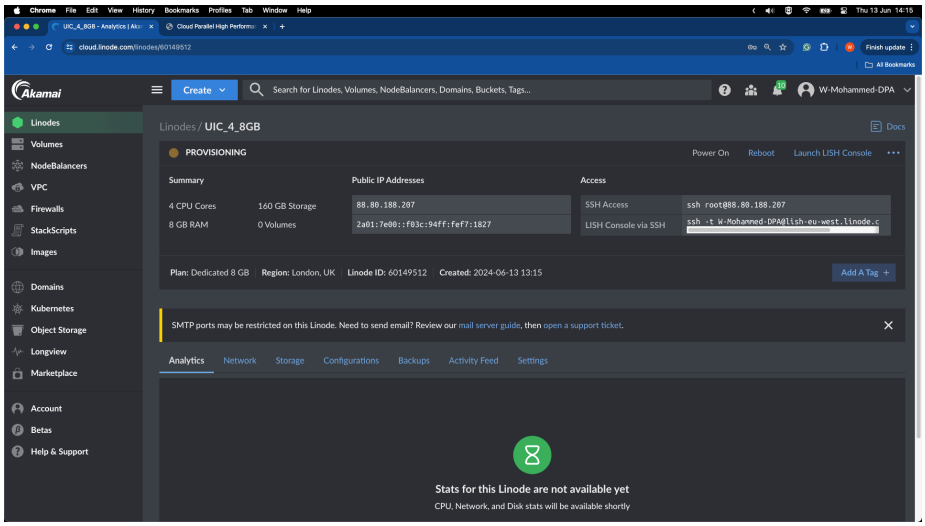

Figure 8: Linode Provisioning

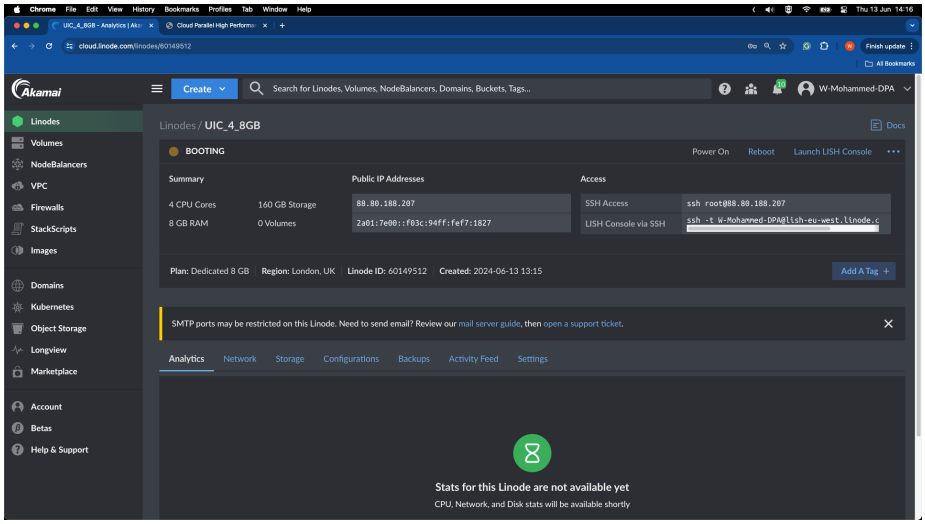

Figure 9: Linode Booting

The Linode instance is ready once it has finished `Booting`; when it is shown that it is `Running`,

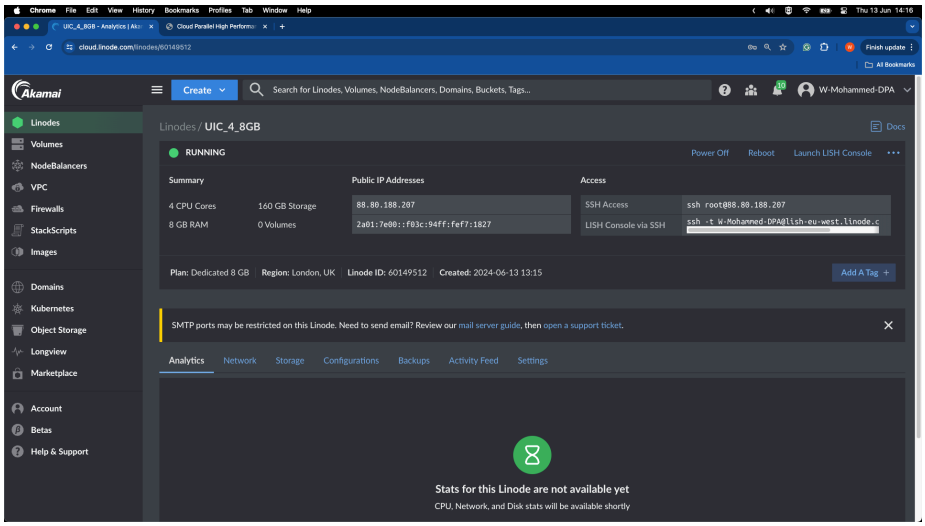

Figure 10: Linode Running

0.4 Accessing a Linode instance:

The Linode instance can be accessed by `ssh`-ing to the instance. Two methods are demonstrated below.

0.4.1 Accessing a Linode instance from the terminal:

- From the Linode dashboard copy the **SSH Access** (`ssh root\@xxx.xxx.xx.xx`), *ssh root@88.80.188.207* for the *UIC_4_8GB*.
- Paste on and execute the command from macOS/Linux terminal.

Figure 11: SSH-ing into the Linode Instance

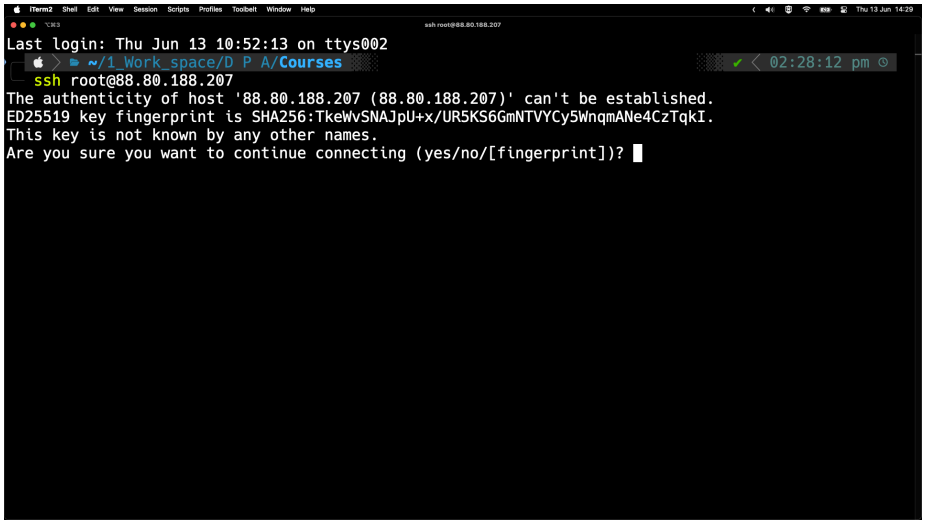

Figure 12: Accept Instance as one of the known hosts

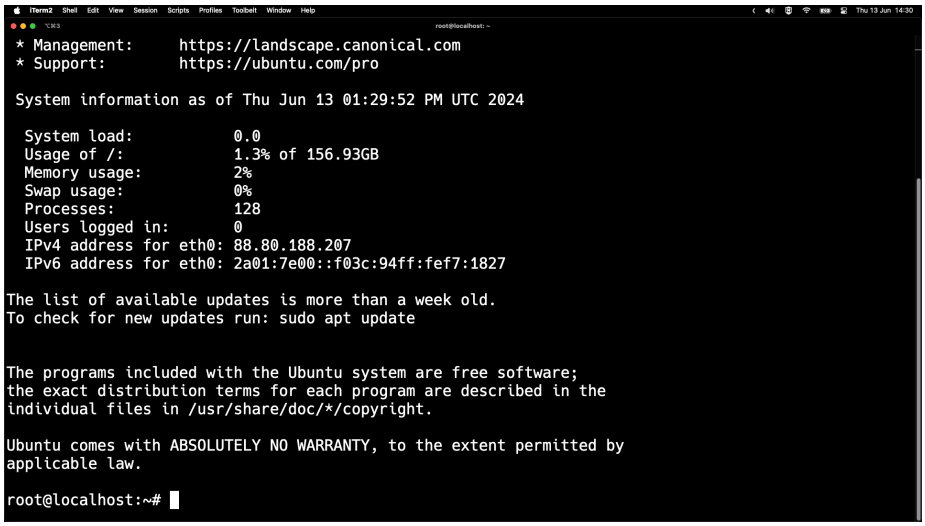

Figure 13: Linode Instance Terminal

The *LISH Console via SSH* allows users to log in to the Linode instance from the Linode SSH Console.

0.5 Moving files to/from the Linode instance:

The following commands are expected to be run from the local machine's terminal, not the Linode instance.

```
# Move one file from the current folder to the root of the instance:
scp cloud_instance_setup.sh root@xxx.xxx.xxx.xxx:

# Move one file from the current folder to a folder 'microSim' located in the root of the ir
scp cloud_instance_setup.sh root@xxx.xxx.xxx.xxx:microSim/

# To move a folder from the current directory to the root of the instance:
scp -r r4he_uic/ root@xxx.xxx.xxx.xxx:
```

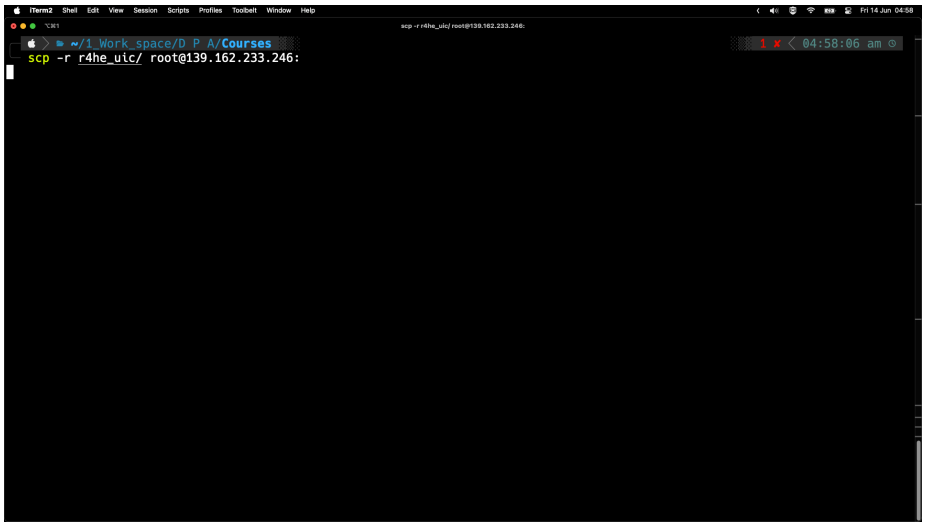

Figure 14: Copy to Linode

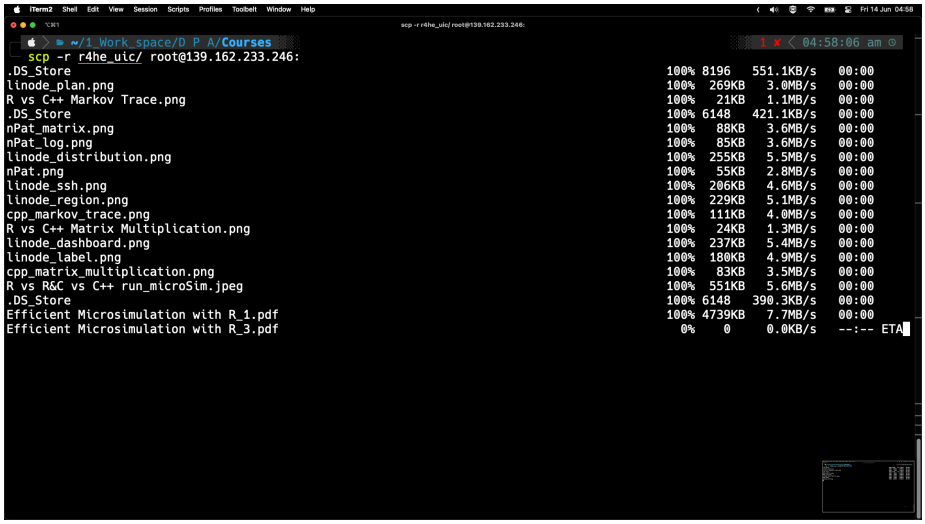

Figure 15: Copy to Linode

```
# To move a file from the Linode instance to our local computer:
scp root@xxx.xxx.xxx.xxx:file_name .
```

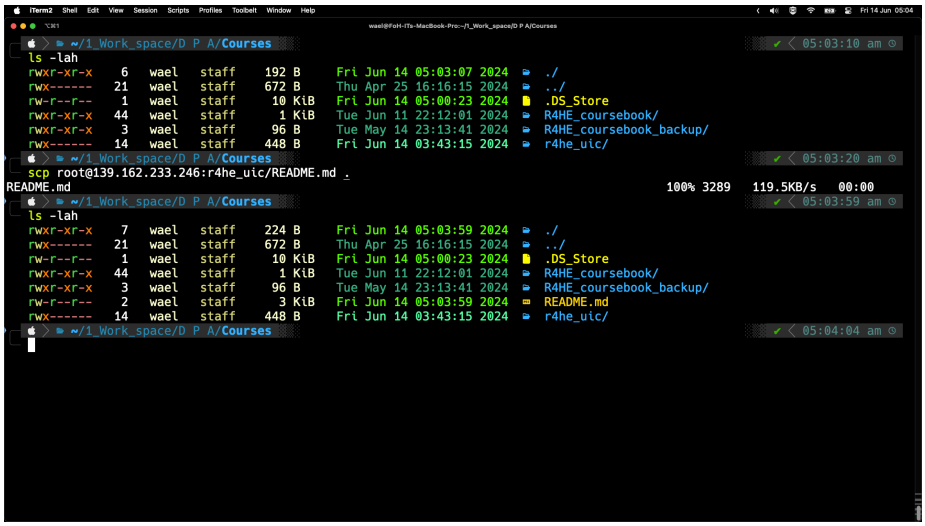

Figure 16: Copy from Linode

To learn more about moving files from and to a remote instance, check https://www.youtube.com/watch?v=lMC5VNoZFhg&t=105s.

0.6 Working on the Linode instance:

0.6.1 Installing R:

Instructions on how to add the latest R version can be found at https://cran.r-project.org/bin/linux/ubuntu/fullREADME.html.

Run the following code from the Linode instance terminal to install `R` and some of the required `R packages`.

```
# Update and upgrade the system
sudo apt update && apt upgrade -y

# Install software-properties-common to manage repositories
sudo apt install software-properties-common -y

# Update and upgrade the system again after installing the add-apt-repository
sudo apt update && apt upgrade -y

# Add the R repository appropriate for the Ubuntu release $(lsb_release -cs)
sudo add-apt-repository "deb https://cloud.r-project.org/bin/linux/ubuntu $(lsb_release -cs)

# Import the R repository signing key
wget -qO- https://cloud.r-project.org/bin/linux/ubuntu/marutter_pubkey.asc | tee -a /etc/apt
```

```
# Update and upgrade the system again after adding new repository
sudo apt update && apt upgrade -y

# Install R base development package
sudo apt install r-base-dev -y

# Install R packages
R -e "install.packages(c('future', 'purrr', 'furrr', 'ggplot2', 'Rcpp', 'RcppArmadillo', 'he
```

The above commands are saved in the `bash` file `cloud_instance_setup.sh`". The"`cloud_instance_setup.sh`" script allows the execution of all commands by using the command `sudo bash ./cloud_instance_setup.sh`.

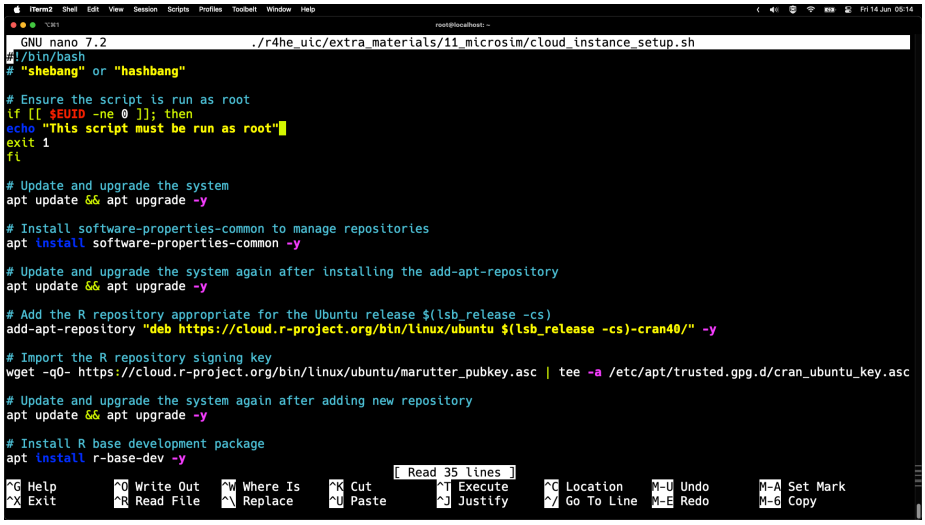

Figure 17: cloud_instance_setup.sh

This command assumes that the `bash` file `cloud_instance_setup.sh` is in the working directory of the Linode instance. See Section 0.5 for how to move files to and from the Linode instance.

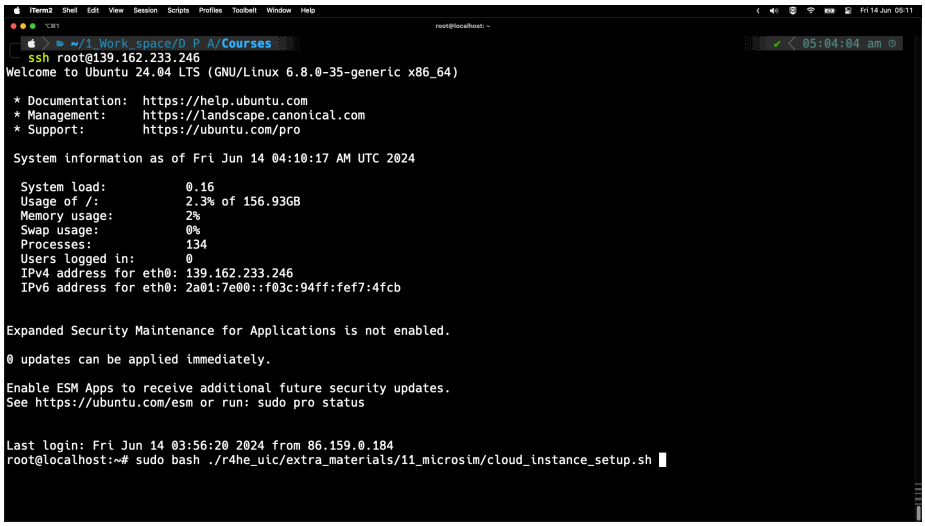

Figure 18: Executing the cloud_instance_setup.sh

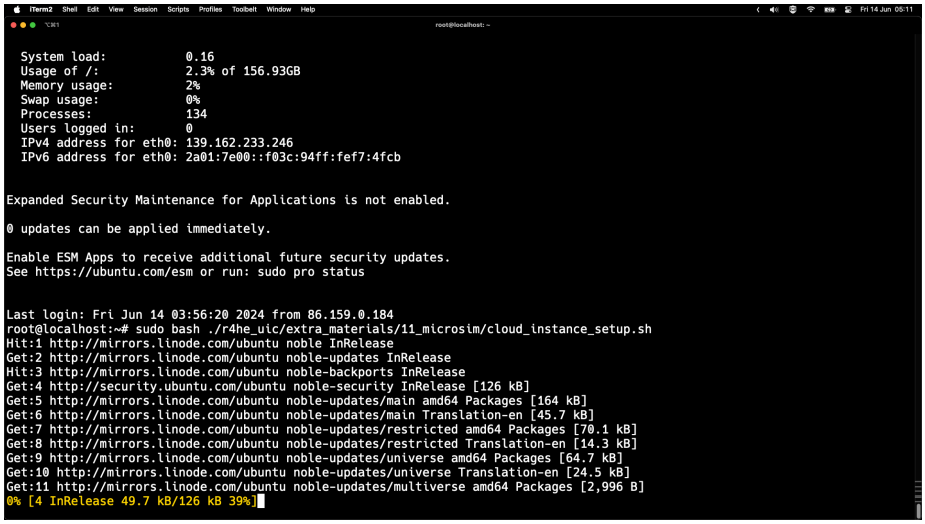

Figure 19: Executing the cloud_instance_setup.sh

0.6.2 Running Rscripts on the Linode instance:

The command `Rscript` is used to execute `R scripts`. The screenshots below showcase running the `R script` that compares parallel execution methods, namely "forking" and "PSOCK". The required code is in the script `cloud_parallel_terminal.R`.

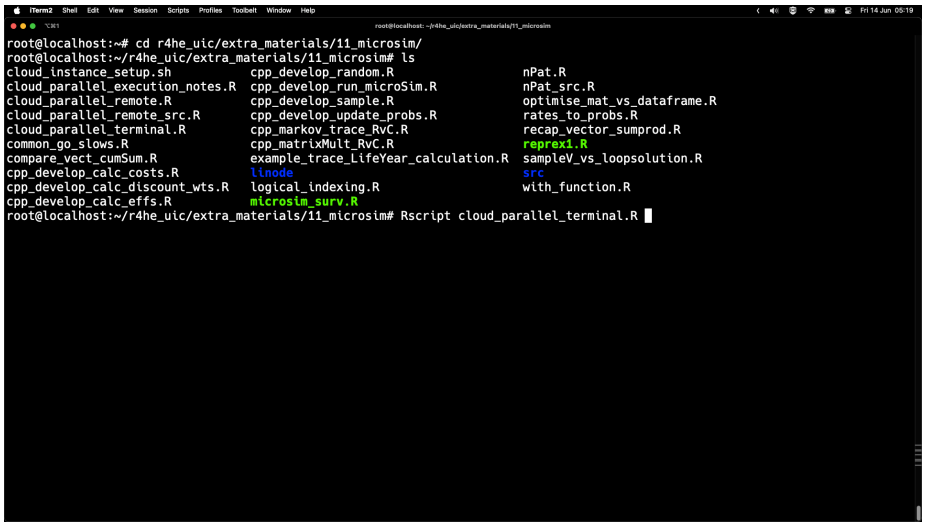

Figure 20: Executing the cloud_instance_setup.sh

The `cloud_parallel_terminal.R` script defines two versions of the `run_psa_parallel()` functions. The first function utilizes the `furrr` and `future` packages; whereas, the second employs the `parallel` function.

The following screenshots represent the outputs from running the `R` script `cloud_parallel_terminal.R`.

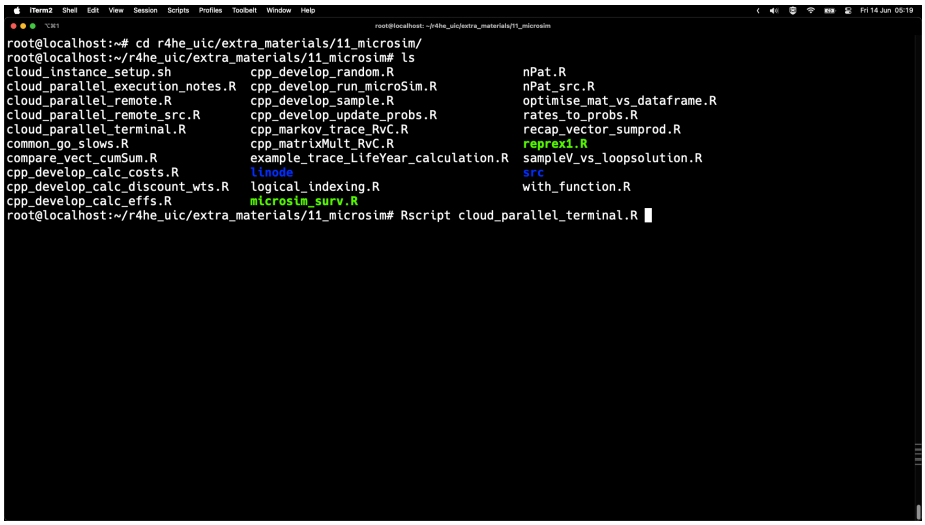

Figure 21: Executing the cloud_instance_setup.sh

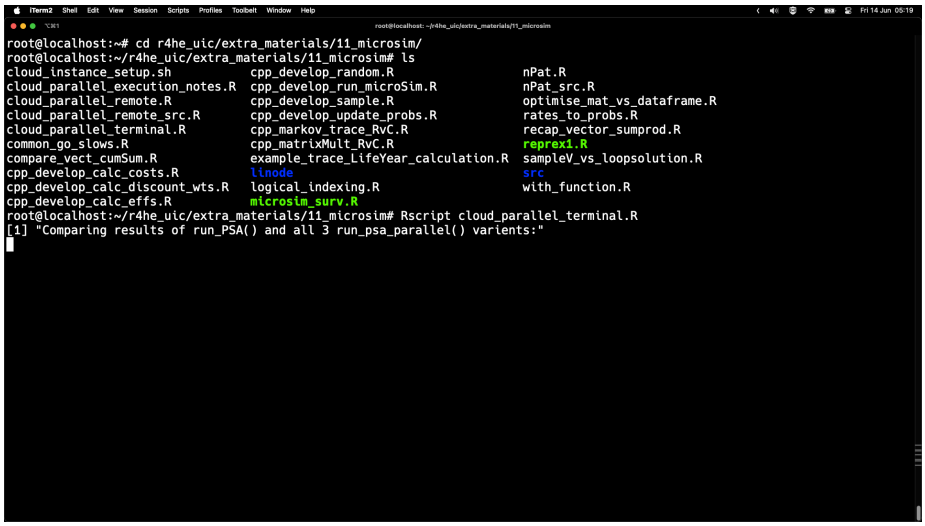

Figure 22: Executing the cloud_instance_setup.sh

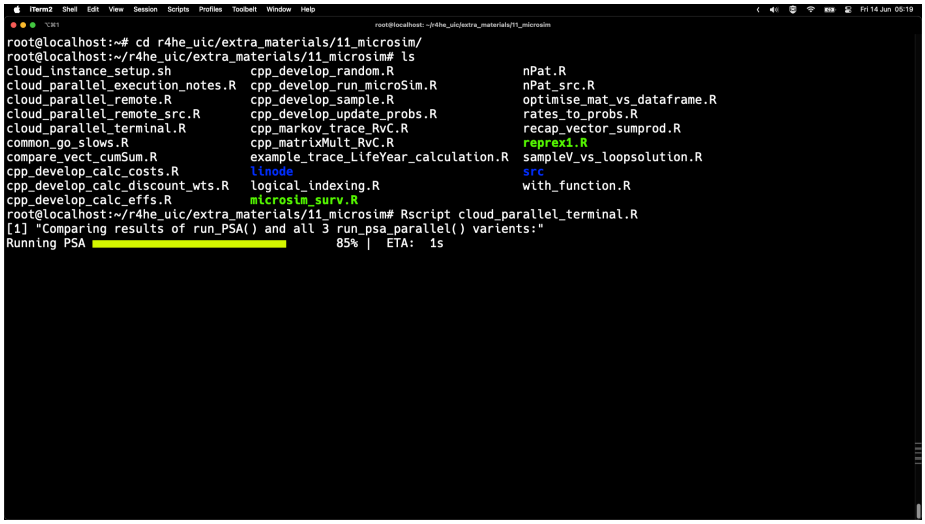

Figure 23: Executing the cloud_instance_setup.sh

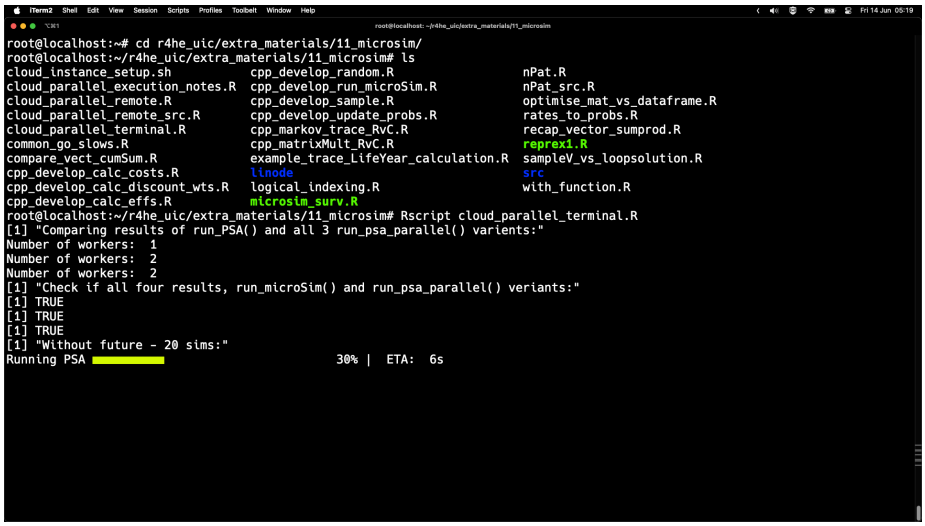

Figure 24: Executing the cloud_instance_setup.sh

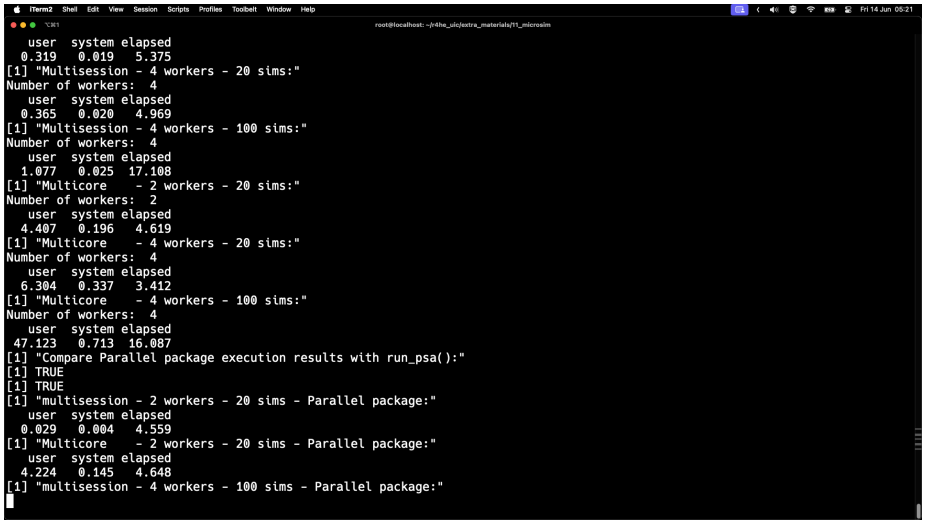

Figure 25: Executing the cloud_instance_setup.sh

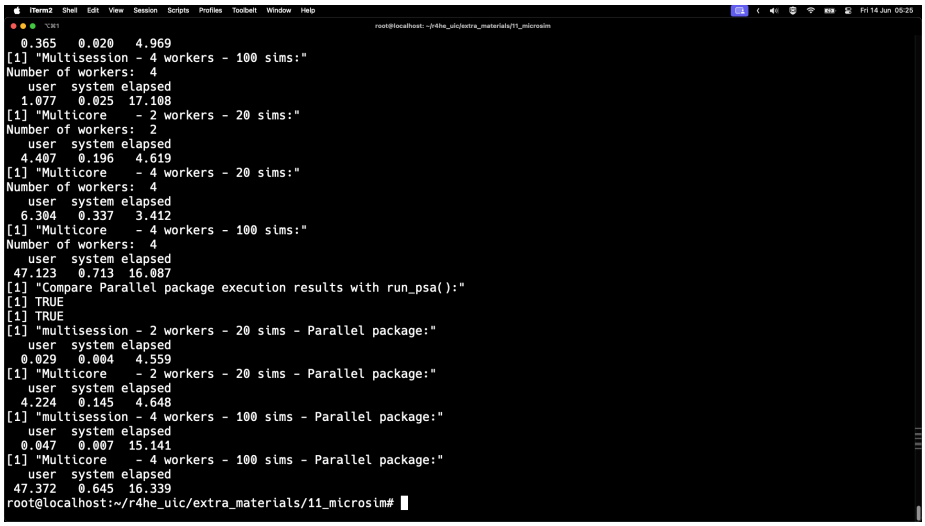

Figure 26: Executing the cloud_instance_setup.sh

0.6.3 Processing local R code remotely without moving any files

This section briefly discusses sending `R` code from an `R` session running on the local machine (the user's laptop or PC) to be executed on a remote cluster. This process does not require the user to interact directly with the remote cluster (moving files to the cluster and running them as in Section 0.5 and Section 0.6.2).

The process is powered by the `furrr` and `future` packages, and the information needed for the `future` package to establish a connection with the cluster is the instance's **`Public IP Address`**.

The code required to demonstrate dispatching and processing code from an `R` session running locally to a Linode instance is described in detail in the `R` script `cloud_parallel_remote.R`.

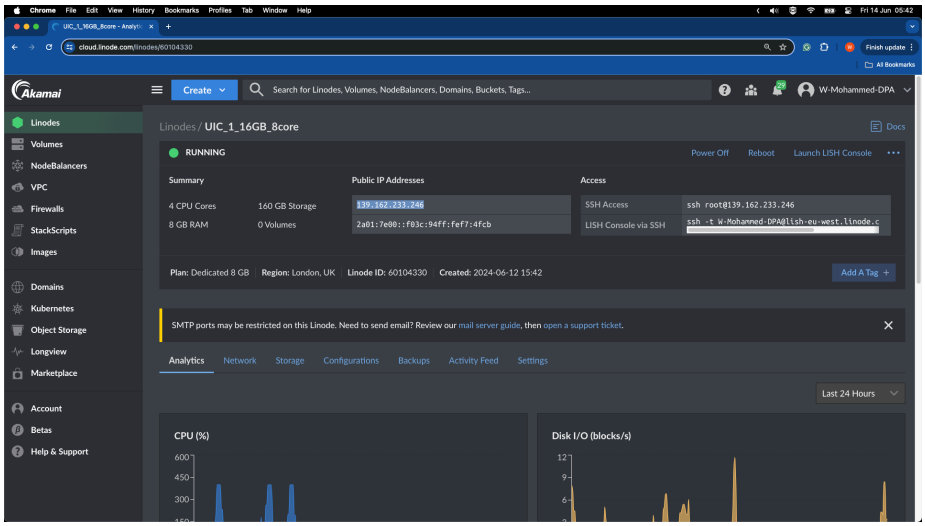

Figure 27: Linode Instance Public IP Address

The command `top -i` or `htop` can be executed on the Linode instance to monitor when or check that the remote execution of `R` code is running on the Linode.

Figure 28: Running the htop command

The `htop` command displays a dynamic task-manager-like interface to show the running processes and their CPU and memory usage, among other statistics.

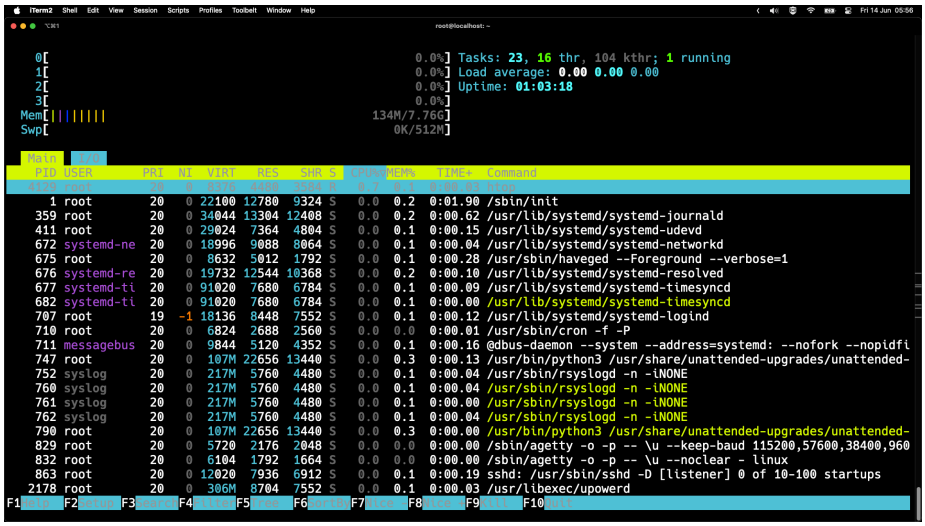

Figure 29: Linode Instance - `htop`

The command `htop` is used below to monitor the execution of the code sent from the `R` script `cloud_parallel_remote.R`. This code involves running `100` PSA iterations on the Linode instance.

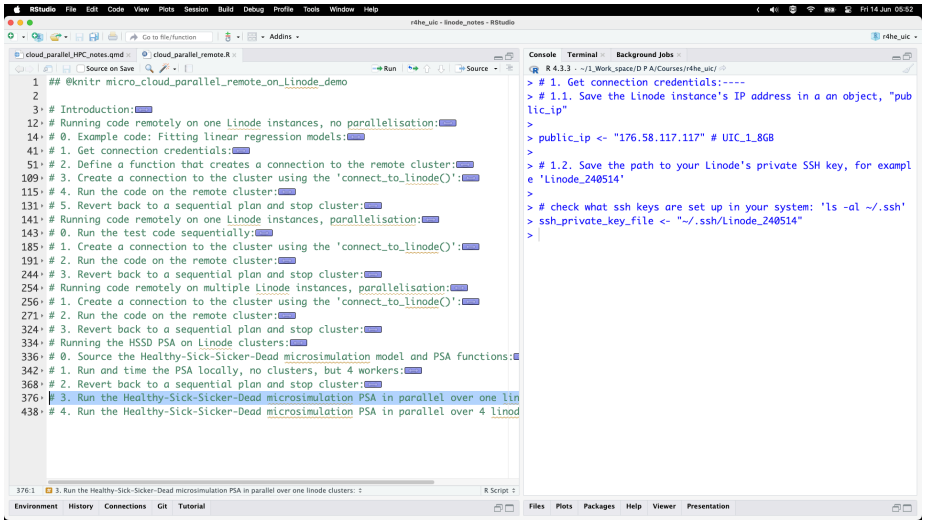

Figure 30: `cloud_parallel_remote.R`

The connection credentials include the SSH private key and the Linode instance Public IP Address.

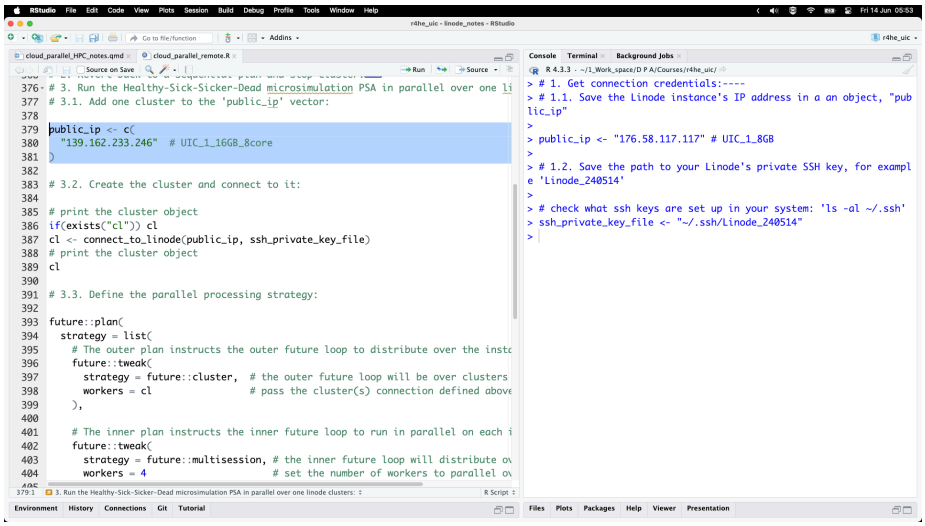

Figure 31: Adding the Linode Instance IP Address

The screenshot below displays the `htop` command (running on the Linode instance terminal) next to the `R` session (running on the local PC). The parallel execution code appears in `R`'s console but has yet to be executed.

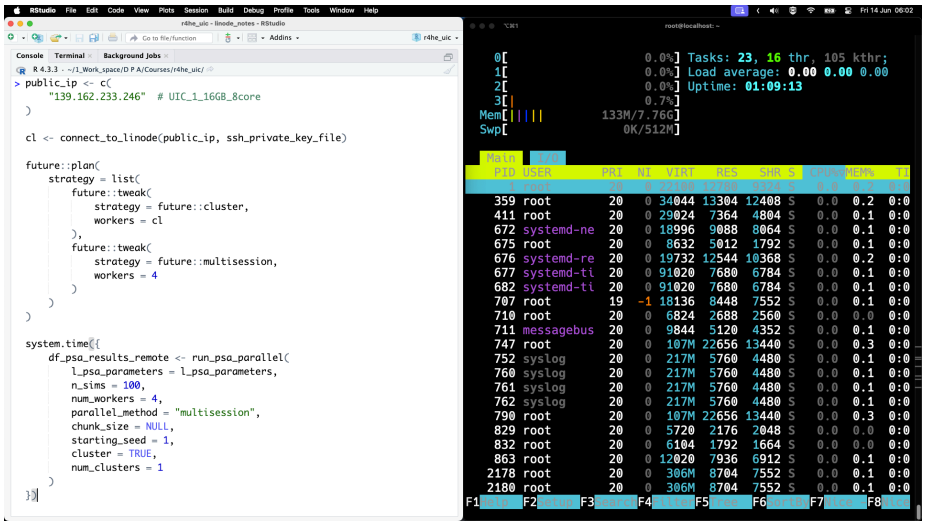

Figure 32: Local `R` Session Vs Remote Linode Instance Terminal

The screenshot below shows the start of the execution of the `R` code. The second line shown in the console (in the screenshot below) represents the part of the

code where `R` establishes a connection with the Linode instance based on its IP Address.

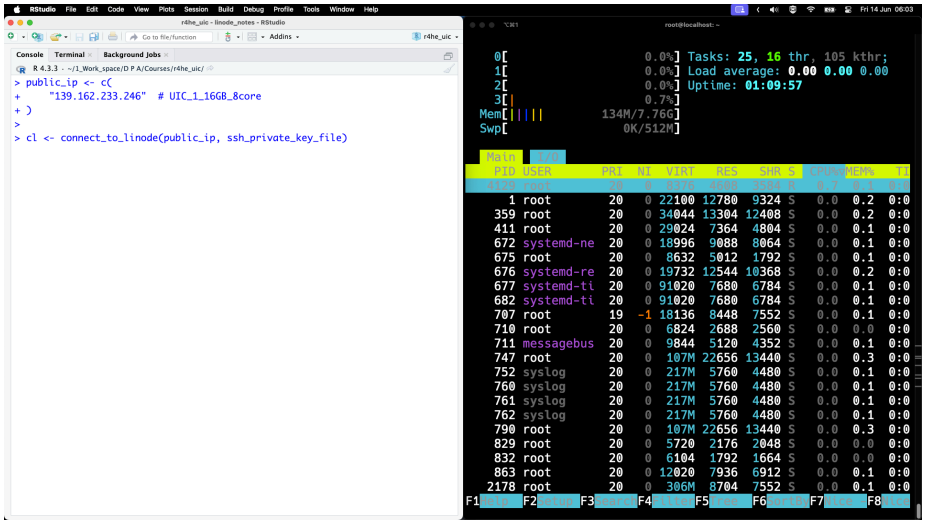

Figure 33: Connecting to the Linode Instance

The parallel execution plan `future::plan`, the code in the console just below the remote connection line, represents the instructions for executing the PSA code on the Linode instance. This plan would apply to multiple clusters if more than one `instance IP Address` is saved in the "`public_ip`" object (each representing a cluster). This point is discussed further and showcased in the `cloud_parallel_remote.R` script.

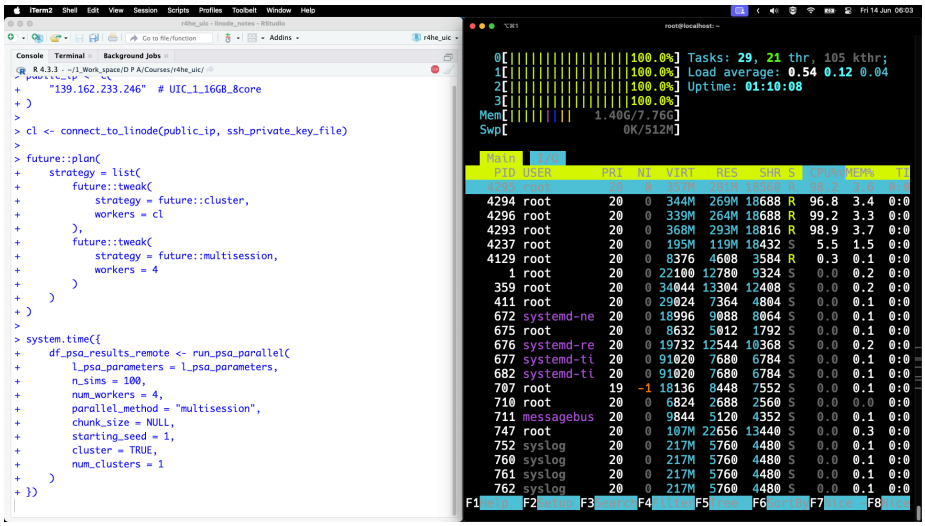

Figure 34: Running Local Code on Remote Instance

The execution of `100` PSA iterations from the local `R` session onto the Linode instance took around `77` seconds.

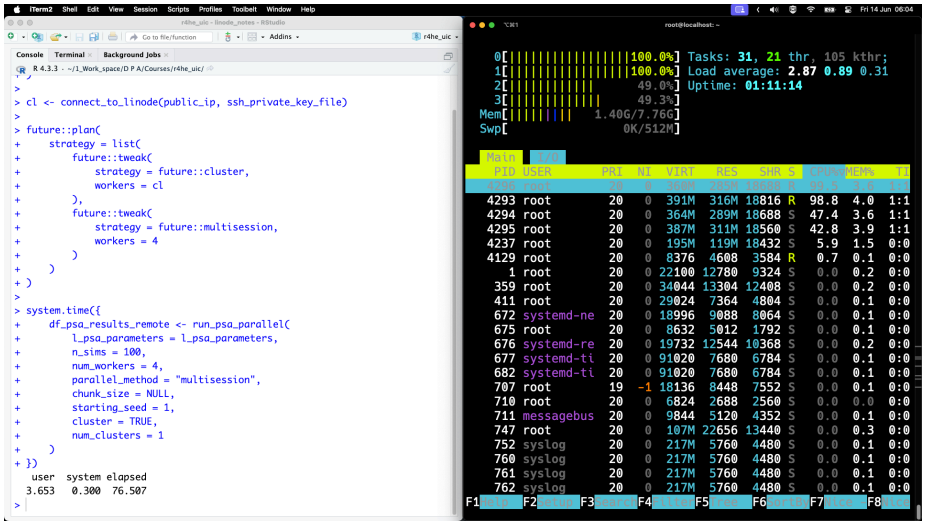

Figure 35: Returning Results from Remote Instance to Local `R` Session

After the parallel code execution finishes, the Linode instance becomes idle, and the `R` processes (running on the remote instance, seen in the `htop` charts in the previous screenshots) are terminated. The' future' package controls the termination of the children `R` processes.

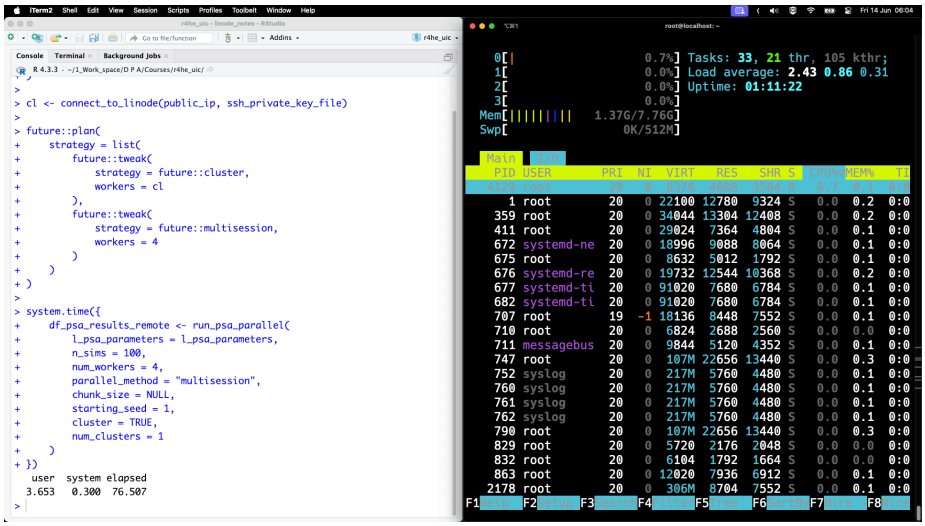

Figure 36: Linode Instance Idle

We hope you find the content useful!

Robert Smith, Wael Mohammed & Paul Schneider