BMRC Setup Guide

Release 0.1

Sansom Lab

Apr 25, 2023

CONTENTS

CHAPTER ONE

INTRODUCTION

1.1 Node CPU architecture

For an overview of the nodes see: https://www.medsci.ox.ac.uk/divisional-services/support-services-1/bmrc/cluster-usage#cluster-queues-and-nodes

Currently, the BMRC cluster is mainly comprised of nodes with “skylake” CPU architecture, but there are also nodes with the older “ivybridge” architecture”. It is important to understand that code compiled on “skylake” will not execute on “ivybridge” CPUs due to additions to the instruction set. Code compiled on “ivybridge” will run on “skylake”.

In the group, we perform our work on skylake login and execution nodes. The two main login nodes, “cluster1.bmrc.ox.ac.uk” and “cluster2.bmrc.ox.ac.uk” both have skylake CPUs. The excution nodes in groups A,E,F also have skylake CPUs.

If, for some reason you need to work from ivybridge, rescomp3 is a hidden ivybridge headnode which can be accessed from the two main login nodes.

1.2 Who is who?

To map the random user names assigned by the BMRC team to real people the following command can be used:

```
ls /well/sansom/users/ | xargs getent passwd | cut -f1,5 -d: | sed 's/:/ -> /g'
```

CHAPTER

TWO

CONNECTING TO THE CLUSTER

2.1 Connecting from within the Kennedy

The BRMC login nodes ("cluster1.bmrc.ox.ac.uk" and "cluster2.bmrc.ox.ac.uk") are directly accessible from the Kennedy network.

2.2 Connecting from outside the Kennedy

Connections from outside the Kennedy should be made via the university VPN. If connecting using an apple mac computer, the CISCO anyconnect client is needed for a stable connection. It can be obtained from the Univerity's IT service.

2.3 Client ssh configuration

An example of a client-side .ssh/config is provided in client-side/ssh/config:

```
Host *
     UseKeychain yes
     AddKeysToAgent yes
     ServerAliveInterval 120

Host bmrc1
     hostname cluster1.bmrc.ox.ac.uk
     user odq545
     ForwardAgent yes
     ForwardX11Trusted yes
     ControlMaster auto
     ControlPath ~/.ssh/sockets/ssh-socket-%r-%h-%p
     ControlPersist 24h

Host bmrc2
     hostname cluster2.bmrc.ox.ac.uk
     user odq545
     ForwardAgent yes
     ForwardX11Trusted yes
     ControlMaster auto
     ControlPath ~/.ssh/sockets/ssh-socket-%r-%h-%p
     ControlPersist 24h
```

Note: You will need to create the ".ssh/sockets" directory if it does not exist:

```
mkdir ~/.ssh/sockets
```

In the example, the "ControlMaster", "ControlPath", and "ControlPersist" directives mean that you should only have to authenticate once a day, with all connections being automatially multiplexed through a single master connection.

CHAPTER

THREE

SETTING UP YOUR BASH ENVIRONMENT

3.1 Removing the login message

Please read the log in messages carefully. Once you have done so, you may not wish to encounter them every time you log in. If so, all you need to do is to touch an empty “.hushlogin” file in you home directory:

```
touch ~/.hushlogin
```

3.2 Configuration

A minimal “.bashrc” file profile is provided in dotfiles/bashrc:

```
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# Detect if we have an interactive shell
if [[ -n "$PS1" ]]; then
    # <--------------- Interactive section ------------------------> #

    # load the group's modules
    if hash module 2> /dev/null ; then
            source /well/sansom/shared/modules/core.sh
            source /well/sansom/shared/modules/single-cell.sh
            source /well/sansom/shared/modules/ngs.sh

            # Remove the scipy bundle from the python path!
            export PYTHONPATH=$(echo ${PYTHONPATH} | awk -v RS=: -v ORS=: '/SciPy/
↪{next} {print}' | sed 's/:*$//')

    fi

    # User specific aliases and functions, e.g.
    alias t='tmux attach || tmux'
    alias s="sqlite3 -header csvdb"
    alias qw="watch qstat -u $USER"
    alias whois="ls /well/sansom/users/ | xargs getent passwd"
```

(continues on next page)

(continued from previous page)

```
    ## example of a function definition to source a python venv
    # venv() {
    #   source ~/devel/venvs/python-3.8.2-GCCcore-9.3.0-skylake/bin/activate
    #   }

    ## SLURM configuration
    export SLURM_CONF=/run/slurm/conf/slurm.conf
    export DRMAA_LIBRARY_PATH=/usr/lib64/libdrmaa.so

    ## Alternative set up for UGE (comment out the two lines above)
    # export DRMAA_LIBRARY_PATH=/mgmt/uge/8.6.8/lib/lx-amd64/libdrmaa.so.1.0

    ## Allow seperate processes to perform concurrent reads from hdf5 files.
    export HDF5_USE_FILE_LOCKING=FALSE

fi
```

Note: It is essential to understand that cluster jobs submitted by CGAT pipelines inherit the bash environment of the main pipeline process. In this scenario, we want bash to initialise without manipulating the environment. To avoid execution of user configuration in this situation, we therefore test whether the "PS1" variable is set in our .bashrc file to determine if the shell is interactive or not. The "PS1" variable is set in interactive shells, but not in non-interactive shells. When you login via ssh, you obtain an interactive shell, the "PS1" variable will be set and configuration code within the if statement will be executed. Pipeline jobs run on execution nodes in non-interactive shells, where "PS1" is not set so the configuration code within the if statement is not executed.

3.3 Software modules

In order for different programs to be able to function together in the same bash environment, generally speaking, they need be compiled with a common toolchain and linked against common libraries.

The Kennedy is currently using the **"gompi/foss-2020a"** and **"GCC/GCCcore-9.3.0"** toolchains which are compatible with each other (the GCC tool chain is effectively a subset of the full toolchain). All the software that you use (both in the shell and elsewere, e.g. Rstudio) should be built with these toolchains or it is very likely that you will encounter errors.

In the group, we load the modules we need via our ".bashrc" in interactive shells as shown above. This is done by sourcing scripts that load sets of related modules. The scripts defining our curated sets of modules are located on the cluster in our "~/shared/modules" folder and are version controlled in this repository. Please submit pull requests to this repo for any changes/updates needed.

Note: It is important that the ~/shared/module/core.sh file is sourced first when loading module sets. It begins by purging the module space to make sure there will be no conflicts and enables use of Kennedy modules built with the current toolchain.

3.4 Using tmux

tmux is an advanced terminal multiplexer. You can use it to keep bash shells open even when you are not logged in. This is very useful for protecting against interruptions to your connection to the cluster. It is recommended to always execute longer running tasks, such as pipelines, from within a tmux session.

An configuration file for tmux that enables mouse support is in dotfiles/tmux:

```
set -g mouse on
```

3.5 Staying under quota in your home directory

By default your home directory has a quota of 10GB. To check the disk usage in your home directory do the following:

```
cd
du -hs
```

It is likely that you will hit the 10GB quota if you use e.g. rstudio, jupyter, bioconductor or other programs that save files into users home folders by default. To avoid this happening, move the problematic folders (typically including ".local", ".cache" and ".jupyter") to a location in your folder in the group's space (/well/sansom/users/$USER/) and then symlink to them from your home folder. The paths should then look like this from your home folder (where $USER is your user name):

```
.cache -> /well/sansom/users/$USER/.cache
.local -> /well/sansom/users/$USER/.local
.jupyter -> /well/sansom/users/$USER/.jupyter
```

CHAPTER

FOUR

WORKING WITH PYTHON

4.1 Python version and module

We are currently using **Python/3.8.2-GCCcore-9.3.0** as our default module.

4.2 Setting up a virtual environment

A Python 3 virtual environment can be set up and activated as follows:

```
#check the version of python3 and the toolchain that you are using
python3 --version

# in this example we are using Python 3.8.2 built with GCCcore-9.3.0

#check the CPU architecture
cat /sys/devices/cpu/caps/pmu_name

# in this example we are using skylake

# make the virtual environment
# it is a good idea to keep track of the python version, tool chain and CPU
# architecture in the name of the folder used for the environment.
# the "--prompt" argument specifies the name that will be displayed
# when the environmet is active.
python3 -m venv --prompt=py382_sky_venv ~/devel/venvs/python-3.8.2-GCCcore-9.3.0-
↪skylake

# activate the virtual environment
source ~/devel/venvs/python-3.8.2-GCCcore-9.3.0-skylake/bin/activate

# update pip
pip install --upgrade pip

# deactivate the environment
deactivate
```

CHAPTER

FIVE

WORKING WITH R

5.1 R module and version

We are currently using the **R/4.1.2-foss-2020a-bare** module as our default.

This module provides R 1.4.2 built with the foss-2020a toolchain with only the base package set. This avoids conflict/confusion between centrally installed libraries and user-installed libraries.

5.2 R configuration

A minimal .Rprofile file is provided in dotfiles/Rprofile:

```
# Set the R package location
# (make sure to edit to your own $USER name!)
.libPaths(c("/well/sansom/users/odq545/devel/R/4.1/skylake/"))

# Set a local mirror for packages
options(repos=structure(c(CRAN="https://www.stats.bris.ac.uk/R/")))

# Set cairo as the default bitmap type
options(bitmapType='cairo')

# Enable vscode hooks
if (interactive() && Sys.getenv("RSTUDIO") == "") {
source(file.path(Sys.getenv(if (.Platform$OS.type == "windows") "USERPROFILE
↪" else "HOME"), ".vscode-R", "init.R"))
}

# End line to ensure that the last command as a new line (else it is not␣
↪executed!)
```

Note: This setup assumes that you will use R only on the skylake nodes. It is different from the set up recommended by the BMRC which performs automatic CPU architecture detection and uses multiple package libraries for different architectures. In the group we prefer to be explicit about the location of the R libraries so that (i) we know which code we are using to perform analysis (important for reproducibility) and (ii) we only install packages once.

Warning: R libraries compiled on skylake will not run on ivybridge nodes. If you need to use ivybridge nodes, you can maintain a seperate pacakage library and switch .libPaths() statements as needed. TODO: investigate use of renv.

CHAPTER

SIX

RUNNING CGAT PIPELINES

6.1 Configuration

The configuration of cgat-core pipelines can be customised via a “.cgat.yml” file in your home directory.

The recommended settings for the BMRC are provided in dotfile/cgat.yml:

```
tmpdir: /tmp
share_tmpdir: /tmp

cluster:
    queue_manager: slurm
    queue: short
    options: "--constraint=skl-compat"
```

With the queue specified here, jobs will be submitted to all the “skylake” nodes by default (i.e. without needing to set the –cluster-queue argument).

6.2 Installing pipelines

Before installing the CGAT code, you first need to set up and activate a Python 3 virtual environment, see: *Working with Python*.

It is then recommended to then install CGAT Core, Apps and Flow into the same (activated) virtual environment using the steps below.

6.2.1 Installing CGAT Core

CGAT core provides a powerful and flexible framework for writing best practise bioinformatics pipelines using Python3 and Ruffus (https://github.com/cgat-developers/ruffus). For more details please read the publication: https://doi.org/10.12688/f1000research.18674.2.

The code is maintained on GitHub here: https://github.com/cgat-developers/cgat-core

Before cloning and setting up the code it is recommended to install the dependencies. These are listed in the repo here: https://github.com/cgat-developers/cgat-core/blob/master/conda/environments/cgat-core.yml . If you skip this step you will need to install any missing packages as you go along.

To clone and setup the repository:

```
# cd to an appropriate location, such as your development folder
cd ~/devel/

# clone the cgat-core repo
git clone git@github.com:cgat-developers/cgat-core.git

# run setup
cd cgat-core/
python setup.py develop
```

6.2.2 Installing CGAT Apps

The CGAT Apps repository provides a collection of scripts for the analysis of high-throughput sequencing data.

As before it is recommended to first install any missing dependencies. These are listed in the repo here: https://github.com/cgat-developers/cgat-apps/blob/master/conda/environments/cgat-apps.yml

To clone and set up the repo:

```
# cd to an appropriate location, such as your development folder
cd ~/devel/

# clone the repo
git clone git@github.com:cgat-developers/cgat-apps.git

# run setup
cd cgat-apps/
python setup.py develop
```

6.2.3 Installing CGAT Flow

CGAT Flow provides a collection of pipelines (written using cgat-core and cgat-apps) for the analysis of next-generation sequencing data such as ChIP-seq, ATAC-seq and RNA-seq data.

Again, it is recommended to first check for and install any missing dependencies. These are listed in the repo here: https://github.com/cgat-developers/cgat-flow/blob/master/conda/environments/cgat-flow-pipelines.yml

To clone and set up the repo:

```
# cd to an appropriate location, such as your development folder
cd ~/devel/

# clone the repo
git clone git@github.com:cgat-developers/cgat-flow.git

# run setup
cd cgat-flow/
python setup.py develop
```

6.3 Known Issues

6.3.1 Internet access

There is currently no internet access on cluster execution nodes. If a job needs internet access, it can be run on a login node by passing the "–no-cluster" argument to the pipeline, e.g.

```
cellhub annotation make full -v5 --no-cluster
```

6.3.2 Excessive waiting for completed jobs

The pipelines wait a specified amount of time before checking whether jobs have completed. Unfortunately, this is controlled by a hardcoded GEVENT setting. This setting was changed from 1 to 30 during the COMBAT project when there were undiagnosed issues with a BMRC-specific DRMAA bug that has subsequently been fixed. To improve pipeline performance it is recommended to edit this setting in your local copy of cgatcore/pipeline/execution.py to e.g.:

```
GEVENT_TIMEOUT_WAIT = 5
```

CHAPTER

SEVEN

WORKING WITH VISUAL STUDIO CODE

Visual Studio Code is a free and powerful IDE that can facilite first-class code editing, shell access, document viewing and more on the cluster. It has very broad adoption in the community and many excellent extensions. It is the recommended way to work on the cluster day-to-day.

7.1 Installation

To get going, download and install Visual Studio Code on your client machine from the link above. You will then need to install the following extensions:

- Remote SSH
- Python for python code sytax highlight and more.
- R for R code syntax highlighting and more.
- vscode-pdf for viewing PDFs
- image preview for viewing images
- Jupyter for jupyter notebooks
- Awesome Emacs Keymap if you are used to emacs keybindings.

7.2 Running R scripts

To evaluate code in rscripts interactively, you will need to have the following in your settings.json file:

```
// Use active terminal for all commands, rather than creating a new R terminal
"r.alwaysUseActiveTerminal": true,

// Use bracketed paste mode
"r.bracketedPaste": true,

// Enable R session watcher
"r.sessionWatcher": true,

// This requires the R httpgd package to be installed
"r.plot.useHttpgd": true,

// Optionally disablye R conding linting:
"r.lintr.enabled": false,
```

And the following lines in your .Rprofile:

```
# Enable vscode hooks
if (interactive() && Sys.getenv("RSTUDIO") == "") {
source(file.path(Sys.getenv(if (.Platform$OS.type == "windows") "USERPROFILE" else
↪"HOME"), ".vscode-R", "init.R"))
}
```

7.3 Syncing settings between machines

Syncing of vscode settings and extensions between machines can be performed easily by linking vscode to your github account as described here.

CHAPTER

EIGHT

HELP

8.1 Sources of information

(1) Kennedy research computing wiki, maintained by Konstantin: http://rcwiki.kennedy.ox.ac.uk

(2) BMRC website: https://www.medsci.ox.ac.uk/divisional-services/support-services-1/bmrc/cluster-usage

8.2 Getting help

In the first instance please contact the Kennedy Institute's dedicated Systems Administrator, Konstantin Borisenko. He is avaliable on Teams or email (konstantin.borisenko@kennedy.ox.ac.uk).

The BRMC help team is also avaliable to help with any issues that might come up, they can be emailed at bmrc-help@medsci.ox.ac.uk.

CHAPTER NINE

INDICES AND TABLES

- genindex
- modindex
- search