Guide to the computing@biostat.ucph

Computer King, 27 October 2020

Abstract

This is an introduction to the computing fascilities at the Section of Biostatistics, Institute of Public Health, University of Copenhagen. It is maintained by Andreas () and the latest version can be found at https://github.com/aejensen/computing-biostat-ucph. This is work in progress.

Contents

1 Introduction

The Section of Biostatistics owns two servers for high performance computing. These are named **cox** (official name: `biostatcomp01fl.sund.root.ku.dk`) and **rao** (official name: `biostatcomp02fl.sund.root.ku.dk`) [It's probably best to use the official names when communicating with KU-IT]. The servers are HPE ProLiant DL325 Gen10 Plus acquired in 2020 and each have an AMD EPYC 7702P processor with 64 cores with 2 threads per core and ½ TB of memory. They run Red Hat Enterprise Linux Server release 7.9 (Maipo) and your personal network drive is mounted as `~` at `/home/ifsv/ku-username`.

Usage of cox and rao is based on submitting jobs to a scheduler. The scheduler used is the the Slurm Workload Manager (https://slurm.schedmd.com/). The following point is extremely important:

↯It is **strictly forbidden** to use cox and rao outside of the Slurm scheduler!

This means that you are **not** allowed to run R interactively on the servers and fork processes using e.g., `mclapply` or other facilities from the parallel package.↯

The **only** exception is if you need to install R packages for subsequent use by jobs submitted to the scheduler. You can do that by starting up R interactively in a terminal on either cox and rao and installing the packages as usual.

The reason behind this strict requirement is that every employee of the section should have equal access to the available resources as fairly as possible. If you are more anarchistically inclined, the section has three older servers that can be used in a free-for-all environment. More on these in section 4 of this document.

2 How to logon

In order to logon to the servers you must first be added as a Linux user on the system. It is *only* KU-IT that can do that, so if you are not already a user on the system, you should contact KU-IT.

If you already have been granted a Linux user, you connect through ssh. You do this from a terminal session on either Windows, OS X and Linux. The username and password is your KU username (in the example below) and password. Example:

```
ComputerKing@hawaii ~ % ssh @cox
Password: ***************
Last login: Fri Oct 16 09:14:27 2020 from 10.34.96.111
Information on usage of the computing cluster and examples of how to submit jobs can be found in
/home/ifsvafd/bsafd/computingcluster and O:\computingcluster. Man pages can be found at.
http://gridscheduler.sourceforge.net/htmlman/manuals.html
bash-4.2$
```

If you are connected to the internet at CSS through the wall, you can connect directly. If you are on eduroam, another WiFi, home, or otherwise outside of CSS you must establish a VPN connection to KU first. See

https://kunet.ku.dk/medarbejderguide/Sider/It/Fjernadgang-vpn.aspx

for information on how to do this.

3 Using the Slurm scheduler on cox and rao

Please remember that

↯It is **strictly forbidden** to use cox and rao outside of the Slurm scheduler!↯

The following commands are used to communicate with the Slurm scheduler and are probably the only ones that you will need.

- `squeue` - view information about jobs located in the Slurm scheduler
- `scancel` - cancel a job running on Slurm
- `sbatch` - submit a command to Slurm

View the status of the scheduler

While logged on to either cox or rao you can view the status of the scheduler by executing the command `squeue` as in the following example

```
bash-4.2$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            2627_0  standard    doSim   xxx999  R       0:24      1 biostatcomp01fl
            2627_1  standard    doSim   xxx999  R       0:24      1 biostatcomp01fl
            2627_2  standard    doSim   xxx999  R       0:24      1 biostatcomp01fl
            2627_3  standard    doSim   xxx999  R       0:24      1 biostatcomp01fl
            2627_4  standard    doSim   xxx999  R       0:24      1 biostatcomp01fl
```

from which it can be seen that the user `xxx999` is running 5 jobs on `biostatcomp01fl` with the name `doSim`, and they have been running for 24 seconds. The job has id 2627 and there are 5 instances running (0, . . . , 5).

Canceling a job on the schedule

To cancel the job `doSim` submitted by the user `xxx999` you can execute the following command

```
scancel 2627
```

You can of course only cancel jobs you have submitted yourself.

Submitting an array job to the scheduler

The must typical use case of cox/rao and the schedule is to perform **array jobs**. This is a type job where you want to perform the same calculations a large number of times. This is typical for e.g., simulation studies of bootstrap calculations.

In the most basic form you submit an array job to cox/rao by using the Slurm command

```
sbatch -a 1-100 -J 'mySimulation' R CMD BATCH myScript.R
```

This will execute the R program in the file `myScript.R` 100 times on cox/rao as a job named `mySimulation` using all available resources.

3.1 A full example

Consider a simple R program where we generate some random data and estimate a parameter

```r
x <- rnorm(1000)
mean(x)
```

```r
number_of_tasks <- as.numeric(Sys.getenv("SLURM_ARRAY_TASK_COUNT"))
task_id <- as.numeric(Sys.getenv("SLURM_ARRAY_TASK_ID"))

# Set a reproducible random seed for your simulations
set.seed(123457890)
# Generate a large number of random seeds to be assigned to each task
seeds <- sample(1:10^6, size = number_of_tasks, replace = FALSE)
# Assign the randomly generated seed to the current task
set.seed(seeds[task_id])

# ---------- Your R program goes between these lines ----------
x <- rnorm(1000)
result <- mean(x)
# -------------------------------------------------------------

# Save the results for this task as an individual file in the output folder
save(result, file = paste('output/result-', task_id, '.RData', sep = ""))
```

4 The older servers running without a scheduler

The Section of Biostatistics still has three older servers: **fisher**, **gauss** and **borel**. You can connect to these through ssh in a similar manner as when connecting to cox and rao.

- **fisher** is from 2013 and is a HP ProLiant BL685c G7 server with 4 AMD Opteron 6380 processors with a total of 32 cores, 64 threads and ½ TB of memory running openSUSE 13.1 (Bottle)
- **gauss** is from 2011 and is an IBM System x3755 M3 server with 4 AMD Opteron 6172 processors with a total of 48 cores and 62.7 GB of memory running Red Hat Enterprise Linux Server release 7.9 (Maipo)
- **borel** is from 2011 and is an IBM System x3755 M3 server with 4 AMD Opteron 6172 processors with a total of 48 cores and 58.8 GB of memory running Red Hat Enterprise Linux Server release 7.9 (Maipo)

These servers are run **without** a workload manager meaning that you can execute programs in anyway that you want e.g., run R interactively and use the multi-core facilities from the parallel package. This implies that you can also **crash** the servers by either submitting too many jobs simultaneously or by exceeding the available memory. Please do not do that.

Crashing the servers implies getting help from KU-IT to restart them, so please **be careful and observant** of other users already running things on these servers. The program `top` which can be executed in the shell is useful to see the current CPU utilization and memory usage and may help you decide whether it is wise to start additional jobs on these servers given their current usage.

It is important to be aware of that due to the different versions of operating systems and gcc not all R packages will work across all the servers if they were install on a specific server. This seems to be especially pronounced for packages depending on Rcpp and its derivatives. Currently, cox and rao are compatible as well as gauss and borel. If an error occurs due to this (most likely seen as a segmentation fault), you can uninstall the packages and reinstall while being logged on to the servers that you subsequently wish to use them on.

Note: It is the plan to shut down gauss and borel by the end of the year 2020.

5 Miscellaneous

The following sections are comments of general usefulness.

5.1 Installation of R packages

You yourself are in control of which packages you have installed in your own local R environment. There are **no** globally or regularly updated R packages installed on the servers. To use some R package you must install them yourself, and they will be downloaded from a CRAN mirror and installed in `~/R`. When R gets updated on the servers you sometimes have to reinstall your R packages.

You can install an R package by starting R interactive and executing the command

```
install.packages("name-of-package-to-install")
```

Note that this is the **only** case where you are allowed to use R on cox and rao interactively outside of the Slurm scheduler.

To make the Danish mirror of CRAN the default repository when installing new packages you can add the following lines to your `~/.Rprofile` file

```
local({
  r <- getOption("repos")
  r["CRAN"] <- "https://mirrors.dotsrc.org/cran"
  options(repos = r)
})
```

Installing some R packages requires compilation of C/C++ code. You can speed-up this process by using multiple cores. To do this you can add `MAKEFLAGS=-j64` to your `~/.R/Makevars` file or more generally add `export MAKEFLAGS=-j64` to your `~/.bash_profile` file.

5.2 GNU Compiler Collection (GCC) on cox and rao

cox and rao are equipped with GCC version 4.8.5 which is the default version on RHEL 7. To use the newest version of GCC as required when e.g., installing the rstan or compiling stan programs you must execute

```
scl enable devtoolset-9 'bash'
```

in your shell. This command will then spawn a new shell with gcc version 9.3.1 as the default compiler. This is **not** persistent between sessions! You can execute `exit` to return to the previous shell with the old version of gcc.

Example:

```
-bash-4.2$ gcc --version
gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44)
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
```

```
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

-bash-4.2$ scl enable devtoolset-9 'bash'
bash-4.2$ gcc --version
gcc (GCC) 9.3.1 20200408 (Red Hat 9.3.1-2)
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

bash-4.2$
```

Note: It's not advisable to install packages while mixing different versions of GCC. This can lead to corruption of your local repository of packages in `~/R`.

5.3 Installing the rstan package

rstan can be a difficult package to install and the requirements have changed through versions. Currently it works on cox and rao if you set the following flags in your `~/.R/Makevars`.

```
CXX14FLAGS=-O3 -march=native -mtune=native -fPIC
CXX14=g++
```

Remember to also enable the 9.3.1 version of GCC before installing rstan. You must also have this enabled in your session when compiling stan scripts.

6 Getting help

If you have small problems or need some software or libraries installed on the servers some of us have `sudo` permissions and might be able to help. You can e.g., contact Andreas (aeje) if you can find him. For larger scale problems or if the servers become non-responsive you should contact KU-IT.