Parallelization in R

Tomáš Martinovič, IT4Innovations, VSB – Technical University of Ostrava

June/2023

Univerza v Ljubljani

TU WIEN TECHNISCHE UNIVERSITÄT WIEN

CINECA

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission.
This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

R parallelization

Tomáš Martinovič, IT4Innovations, VSB – Technical University of Ostrava

- In its raw form R evaluates function in a single process with single thread.
- It may be build with libraries like BLAS, which will enable implicit parallelization when computing vector/matrix operations.
- Another possibility is to run multiple threads at C level via OpenMP or pthreads.
- At the C level we can even make GPU programming possible, but it is actually calling C functions from R.
- Parallelization in R depends on running multiple processes as R interpreter is not thread-safe.

- There are three main ways of creating processes in R:
  1. Spawning new processes and then take care of communication, which is usually done through sockets. This is available on all systems, but firewalls may ask to allow communication.
  2. Forking processes. Fork is a complete copy of the current process including its workspace. Forks should share memory pages and thus is very fast. This does not work on Windows systems.
  3. Using another facilities like MPI. New solutions could be based on different message brokers such as RabbitMQ or ZeroMQ.

R process spawning

Tomáš Martinovič, IT4Innovations, VSB – Technical University of Ostrava

- There are two main ways how to ways to spawn clusters:
  - makePSOCKcluster()
    - Works everywhere, but generally is a little bit slower.
  - makeForkCluster()
    - Works on POSIX based OS.
- Related to that is how to call parallel computation using parallel apply family replacements
  - parLapply(cl, x, FUN, …)
    - Depends on cluster provided by cl.
  - mclapply(X, FUN, …, mc.cores)
    - Works on POSIX based OS.

Mandelbrot set computation

Tomáš Martinovič, IT4Innovations, VSB – Technical University of Ostrava

- Mandelbrot set is based upon computation of

$$f_c(z) = z^2 + c$$

We are iterating this function, by using the last Iteration result as an input for the new iteration. When the distance of z from 0 is greater than 2 We know it will not converge and note how fast it overcome 2.

```
mandelbrot <- function(c, max.iter = 100) {
  z <- 0
  for (i in 1:max.iter) {
    z <- z^2 + c
    if (abs(z) > 2) {
      return(i)
    }
  }
  return(0)
}
```

Mandelbrot set computation 2

Tomáš Martinovič, IT4Innovations, VSB – Technical University of Ostrava

- This computation is then made on a grid going from -2 to 1 on x-axis and -1.5 to 1.5 on y-axis.
- Values we get for each cell are then visualized or print the Matrix values.
- It is easy to parallelize computation of Mandelbrot set, since each process is independent.

```
resolution <- 100
max.iter <- 100

x <- seq(-2, 1, length.out = resolution)
y <- seq(-1.5, 1.5, length.out = resolution)
points <-
  outer(x, y, function(x, y)
    complex(real = x, imaginary = y))
```

```
# apply
apply_fun =  apply(
      X = points,
      MARGIN = c(1, 2),
      FUN = mandelbrot
    )
# for loop
for_loop = for_loop(points)
# parallel apply
parApply(
  cl,
  X = points,
  MARGIN = c(1, 2),
  FUN = mandelbrot
),
```

Tomáš Martinovič, IT4Innovations, VSB – Technical University of Ostrava

- The most used way of computation on HPC cluster for multimode with R is still usage of MPI.
- In case of machine learning jobs it is usually enough to use distribute the computation of the same problem on different data or hyperparameter settings, or different algorithms on the same data.
- This can be achieved in two ways:
  - The first is to use parallel version of apply family.
  - The second is to compute the chunks by hand and gather results to the main rank.

Tomáš Martinovič, IT4Innovations, VSB – Technical University of Ostrava

- When using R with MPI on Karolina cluster it is necessary to
  - Copy the communicator using invisible(mpi.comm.dup(0, 1)) at the beginning of the computation
  - Properly exit MPI with mpi.exit()

Resources

Tomáš Martinovič, IT4Innovations, VSB – Technical University of Ostrava

- https://stat.ethz.ch/R-manual/R-devel/library/parallel/doc/parallel.pdf
- https://cran.r-project.org/web/packages/Rmpi/Rmpi.pdf

Thank you for your attention!

http://sctrain.eu/

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission.
This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.