R with future - Using `future.apply` and `doFuture`

Fabian Freund, Brigitte Wellenkamp
(KIM Hohenheim, bwHPC)

July 26th, 2022
ZOOM

UNIVERSITY OF HOHENHEIM
bw HPC

Using futures in code

- We have seen the main functionality of futures
- Probably, we simply would like to run any function `f` executed within a loop as `future(f)` and return its value
- `future.apply` does this for `*apply` commands and `doFuture` for the `foreach` approach

In case not yet loaded

```
library(doFuture)
```

```
## Loading required package: foreach
## Loading required package: future
```

```
library(future.apply)
```

Setting up plan & a simple function to illustrate

```
plan(strategy = multisession, workers=3)

f_w_il <- function(i){pids <- rep(0,6)
                      for (j in 1:6){
                      Sys.sleep(1)
                      pids[j] <- Sys.getpid()
                      }
                      return(list(job=i,Rproc=table(pids)))}
```

Use a function with an inner loop

A first use of `future.*apply`

```
future_lapply(1:10,f_w_il)
str(future_lapply)
?future_lapply
```

- What is enveloped within a single future? The instances of the function called? What about the inner loop?
- Let's discuss load balancing (feel free to tweak the code)

Quite simple: doFuture

```
plan(multisession,workers=3)
registerDoFuture() #to register
                   #the parallel backend
foreach(i=1:10,.combine=cbind) %dopar% {f_w_il(i)}
foreach(i=1:10,
        .options.future = list(chunk.size = 5)) %dopar% {
          f_w_il(i)}
```

- Works essentially as `foreach`
- doFuture takes care of exports and packages automatically, analogously to `future` (details here)
- Does not allow directly to use parallel seeds. Use library(doRNG) for this
- See `?doFuture` for more options/details

A function that actually does something and takes some time

```
simulate_sth <- function(parameters=c(5000,1,1,
                                      1,10000)){
out1 <- 0
for (i in 1:parameters[5]){
howmany <- rpois(1,parameters[1])
random_weights <- rexp(howmany,parameters[2])
rvs <- rnorm(n = howmany,mean = parameters[3],
             sd = parameters[4])
out1 <-out1 + sum(random_weights*rvs)
}
return(out1/(parameters[5]*parameters[1]))
}
```

- What does the function do?
- Simulate 10 values using a) `future.apply` b) `doFuture`