Gmail by Google

David B. Dahl <dahl@stat.byu.edu>

Seeking your help on issues with rscala

David B. Dahl <dahl@stat.byu.edu> Thu, Jul 26, 2018 at 10:01 AM
To: Hanyu Song <hanyu.song@duke.edu>

Hanyu,

Yesterday I wrote, "I suppose that each R thread/process could instantiate its own bridge" but in encouraged you to use the "multicore" argument instead for parallelization. If, however, you do want to use the parallel package --- because, for example, the salso function is being used in a larger context --- you can do so as illustrated in this code:

```
library(parallel)
load('psm_ex.RData')

cl <- makeCluster(4, type = 'FORK')
clusterEvalQ(cl,library(sdols))

### IMPORTANT: You need to call library(sdols) in clusterEvalQ **BEFORE**
doing
### so in the main script.  But you don't need to call it in the main script,
### unless you want to run your check.

# Check whether this runs
library(sdols)
salso(psm, loss = 'lowerBoundVariationOfInformation')

res <- parLapply(cl, 1:10, function(i) salso(psm, loss = '
lowerBoundVariationOfInformation', multicore=FALSE))

stopCluster(cl)
```

Note that, in this case, it makes sense to use the argument multicore=FALSE since the parallelization is provided by the parallel package and having the salso function do parallelization on top of that will likely slow everything down a bit.

-- David

On Wed, Jul 25, 2018 at 11:41 AM David B. Dahl <dahl@stat.byu.edu> wrote:

> Hanyu,
>
> The sdols package is implemented using the rscala package, which provides a bridge between Scala and R. The bridge is not thread-safe, meaning that multiple R threads or processes cannot access the same bridge simultaneously. I suppose that each R thread/process could instantiate its own bridge, but I'm not sure you want to bother. The salso function has the argument "multicore" which defaults to TRUE hence multiple cores will be used behind the scenes. Please let me know if this doesn't resolve your issue (or if you have any other issues).
>
> By the way, if you are using the latest versions of sdols and rscala on CRAN, you probably do not need to set the heap maximum. (Your earlier emails helped to motivate me to find a better default.) If you do

> want to set it, you should now use something like this: rscala::scalaHeapMaximum("2G").
>
> -- David