

Parallel computation in R

Robert Stojnic rs550@cam.ac.uk

Laurent Gatto lg390@cam.ac.uk

November 4, 2014

Parallel execution

Introduction

- ▶ Applicable when repeating *independent* computations a certain number of times; results just need to be combined after parallel executions are done.
- ▶ A cluster of nodes: generate multiple workers listening to the master; these workers are new processes that can run on the current machine or a similar one with an identical R installation. Should work on all R platforms (as in package *snow*).
- ▶ The R process is *forked* to create new R processes by taking a complete copy of the masters process, including workspace (pioneered by package *multicore*). Does not work on Windows.
- ▶ Grid computing.

Parallel execution

Packages

- ▶ Package *parallel*, first included in R 2.14.0 builds on CRAN packages *multicore* and *snow*.
- ▶ Package *foreach*, introducing a new looping construct supporting parallel execution. Natural choice to parallelise a `for` loop.

parallel example

Dropin replacement for `*apply` functions

- ▶ `mclapply(X, FUN, ...)` (adapted from *multicore*).
- ▶ `parLapply(cl, X, FUN, ...)` (adapted from *snow*).

(demonstration)

foreach example

```
library(doMC)
library(foreach)
registerDoMC(2)
foreach(i = 1:10) %dopar% f(i)
foreach(i = 1:10) %do% f(i) ## serial version

library(plyr)
llply(1:10, f, .parallel=TRUE)
```

(demonstration)

In Bioconductor

BiocParallel

This package provides modified versions and novel implementation of functions for parallel evaluation, tailored to use with Bioconductor objects.

```
library("BiocParallel")

ll <- replicate(8, matrix(rnorm(1e6), 1000),
                        simplify=FALSE)
f <- function(x) mean(solve(x), trim=0.7)

res <- bplapply(ll, f) ## will use the default parallel backend
unlist(res)

## [1] -2.937504e-05 -3.097237e-05 -6.520130e-05  7.443127e-05
## [6] -6.234002e-05 -3.949962e-05 -3.889565e-05
```

MulticoreParam()

```
## class: MulticoreParam; bpisup: TRUE; bpworkers: 4; catch.errors: TRUE  
## setSeed: TRUE; recursive: TRUE; cleanup: TRUE; cleanupSignal: 15;  
## verbose: FALSE
```

SnowParam()

```
## class: SnowParam; bpisup: FALSE; bpworkers: 4; catch.errors: TRUE  
## cluster spec: 4; type: PSOCK
```

SnowParam(type = "MPI")

```
## class: SnowParam; bpisup: FALSE; bpworkers: 4; catch.errors: TRUE  
## cluster spec: 4; type: MPI
```

SerialParam()

```
## class: SerialParam; bpisup: TRUE; bpworkers: 1; catch.errors: TRUE
```

```
p <- MulticoreParam(2L)
unlist(bplapply(ll, f, BPPARAM=p))

## [1] -2.937504e-05 -3.097237e-05 -6.520130e-05  7.443127e-05
## [6] -6.234002e-05 -3.949962e-05 -3.889565e-05

p <- SnowParam(4L)
unlist(bplapply(ll, f, BPPARAM=p))

## [1] -2.937504e-05 -3.097237e-05 -6.520130e-05  7.443127e-05
## [6] -6.234002e-05 -3.949962e-05 -3.889565e-05
```


References

Further reading

- ▶ Parallel R, McCallum and Weston, O'Reilly (2011).
- ▶ *parallel* and *foreach* vignettes.
- ▶ *High Performance Computing* CRAN task view.