Introduction to *BatchtoolsParam*

Nitesh Turaga[1]***, Martin Morgan***[2]

[1]Nitesh.Turaga@RoswellPark.org

[2]Martin.Morgan@RoswellPark.org

Edited: March 22, 2018; Compiled: August 22, 2021

Contents

1 Introduction

The `BatchtoolsParam` class is an interface to the *batchtools* package from within *BiocParallel*. This aims to replace BatchjobsParam as *BiocParallel*'s class for computing on a high performance cluster such as SGE, TORQUE, LSF, SLURM, OpenLava.

2 Quick start

This example demonstrates the easiest way to launch a 100000 jobs using batchtools. The first step involves creating a `BatchtoolsParam` class. You can compute using 'bplapply' and then the result is stored.

```
library(BiocParallel)

## Pi approximation
piApprox <- function(n) {
    nums <- matrix(runif(2 * n), ncol = 2)
    d <- sqrt(nums[, 1]^2 + nums[, 2]^2)
    4 * mean(d <= 1)
}

piApprox(1000)
```

```
## [1] 3.144
```

```
## Apply piApprox over
param <- BatchtoolsParam()
result <- bplapply(rep(10e5, 10), piApprox, BPPARAM=param)
mean(unlist(result))
```

```
## [1] 3.141422
```

3 *BatchtoolsParam* interface

The `BatchtoolsParam` interface allows you to replace the BatchJobsParam interface, and allow more intuitive usage of your high performance cluster with *BiocParallel*.

The `BatchtoolsParam` class allows the user to specify many arguments to customize their jobs. Applicable to clusters with formal schedulers.

- `workers` The number of workers used by the job.
- `cluster` We currently support, SGE, SLURM, LSF, TORQUE and OpenLava. The 'cluster' argument is supported only if the R environment knows how to find the job scheduler. Each cluster type uses a template to pass the job to the scheduler. If the template is not given we use the default templates as given in the 'batchtools' package. The cluster can be accessed by 'bpbackend(param)'.
- `registryargs` The 'registryargs' argument takes a list of arguments to create a new job registry for you `BatchtoolsParam`. The job registry is a data.table which stores all the required information to process your jobs. The arguments we support for registryargs are:

  `file.dir` Path where all files of the registry are saved. Note that some templates do not handle relative paths well. If nothing is given, a temporary directory will be used in your current working directory.

  `work.dir` Working directory for R process for running jobs.

  `packages` Packages that will be loaded on each node.

  `namespaces` Namespaces that will be loaded on each node.

  `source` Files that are sourced before executing a job.

  `load` Files that are loaded before executing a job.

```
registryargs <- batchtoolsRegistryargs(
    file.dir = "mytempreg",
    work.dir = getwd(),
    packages = character(0L),
    namespaces = character(0L),
    source = character(0L),
    load = character(0L)
)
param <- BatchtoolsParam(registryargs = registryargs)
param
```

```
## class: BatchtoolsParam
##   bpisup: FALSE; bpnworkers: 4; bptasks: 0; bpjobname: BPJOB
##   bplog: FALSE; bpthreshold: INFO; bpstopOnError: TRUE
```

```
##   bpRNGseed: NA; bptimeout: 2592000; bpprogressbar: FALSE
##   bpexportglobals: TRUE
##   bplogdir: NA
##   bpresultdir: NA
##   cluster type: multicore
##   template: NA
##   registryargs:
##     file.dir: mytempreg
##     work.dir: /tmp/RtmpqN763R/Rbuild261f72732c1e6e/BiocParallel/vignettes
##     packages: character(0)
##     namespaces: character(0)
##     source: character(0)
##     load: character(0)
##     make.default: FALSE
##   saveregistry: FALSE
##   resources:
```

- `resources` A named list of key-value pairs to be subsituted into the template file; see `?batchtools::submitJobs`.
- `template` The template argument is unique to the `BatchtoolsParam` class. It is required by the job scheduler. It defines how the jobs are submitted to the job scheduler. If the template is not given and the cluster is chosen, a default template is selected from the batchtools package.
- `log` The log option is logical, TRUE/FALSE. If it is set to TRUE, then the logs which are in the registry are copied to directory given by the user using the `logdir` argument.
- `logdir` Path to the logs. It is given only if `log=TRUE`.
- `resultdir` Path to the directory is given when the job has files to be saved in a directory.

4 Defining templates

The job submission template controls how the job is processed by the job scheduler on the cluster. Obviously, the format of the template will differ depending on the type of job scheduler. Let's look at the default SLURM template as an example:

```
fname <- batchtoolsTemplate("slurm")

## using default 'slurm' template in batchtools.

cat(readLines(fname), sep="\n")

## #!/bin/bash
##
## ## Job Resource Interface Definition
## ##
## ## ntasks [integer(1)]:       Number of required tasks,
## ##                            Set larger than 1 if you want to further parallelize
## ##                            with MPI within your job.
## ## ncpus [integer(1)]:        Number of required cpus per task,
## ##                            Set larger than 1 if you want to further parallelize
## ##                            with multicore/parallel within each task.
```

```
## ## walltime [integer(1)]:     Walltime for this job, in seconds.
## ##                           Must be at least 60 seconds for Slurm to work properly.
## ## memory   [integer(1)]:     Memory in megabytes for each cpu.
## ##                           Must be at least 100 (when I tried lower values my
## ##                           jobs did not start at all).
## ##
## ## Default resources can be set in your .batchtools.conf.R by defining the variable
## ## 'default.resources' as a named list.
##
## <%
## # relative paths are not handled well by Slurm
## log.file = fs::path_expand(log.file)
## -%>
##
##
## #SBATCH --job-name=<%= job.name %>
## #SBATCH --output=<%= log.file %>
## #SBATCH --error=<%= log.file %>
## #SBATCH --time=<%= ceiling(resources$walltime / 60) %>
## #SBATCH --ntasks=1
## #SBATCH --cpus-per-task=<%= resources$ncpus %>
## #SBATCH --mem-per-cpu=<%= resources$memory %>
## <%= if (!is.null(resources$partition)) sprintf(paste0("#SBATCH --partition='", resources$partition, "'")) %>
## <%= if (array.jobs) sprintf("#SBATCH --array=1-%i", nrow(jobs)) else "" %>
##
## ## Initialize work environment like
## ## source /etc/profile
## ## module add ...
##
## ## Export value of DEBUGME environemnt var to slave
## export DEBUGME=<%= Sys.getenv("DEBUGME") %>
##
## <%= sprintf("export OMP_NUM_THREADS=%i", resources$omp.threads) -%>
## <%= sprintf("export OPENBLAS_NUM_THREADS=%i", resources$blas.threads) -%>
## <%= sprintf("export MKL_NUM_THREADS=%i", resources$blas.threads) -%>
##
## ## Run R:
## ## we merge R output with stdout from SLURM, which gets then logged via --output option
## Rscript -e 'batchtools::doJobCollection("<%= uri %>")'
```

The `<%= =>` blocks are automatically replaced by the values of the elements in the `resources` argument in the `BatchtoolsParam` constructor. Failing to specify critical parameters properly (e.g., wall time or memory limits too low) will cause jobs to crash, usually rather cryptically. We suggest setting parameters explicitly to provide robustness to changes to system defaults. Note that the `<%= =>` blocks themselves do not usually need to be modified in the template.

The part of the template that is most likely to require explicit customization is the last line containing the call to `Rscript`. A more customized call may be necessary if the R installation is not standard, e.g., if multiple versions of R have been installed on a cluster. For example, one might use instead:

```
echo 'batchtools::doJobCollection("<%= uri %>")' |\
```

```
ArbitraryRcommand --no-save --no-echo
```

If such customization is necessary, we suggest making a local copy of the template, modifying it as required, and then constructing a `BiocParallelParam` object with the modified template using the `template` argument. However, we find that the default templates accessible with `batchtoolsTemplate` are satisfactory in most cases.

5 Use cases

As an example for a BatchtoolParam job being run on an SGE cluster, we use the same `piApprox` function as defined earlier. The example runs the function on 5 workers and submits 100 jobs to the SGE cluster.

Example of SGE with minimal code:

```
library(BiocParallel)

## Pi approximation
piApprox <- function(n) {
    nums <- matrix(runif(2 * n), ncol = 2)
    d <- sqrt(nums[, 1]^2 + nums[, 2]^2)
    4 * mean(d <= 1)
}

template <- system.file(
    package = "BiocParallel",
    "unitTests", "test_script", "test-sge-template.tmpl"
)
param <- BatchtoolsParam(workers=5, cluster="sge", template=template)

## Run parallel job
result <- bplapply(rep(10e5, 100), piApprox, BPPARAM=param)
```

Example of SGE demonstrating some of `BatchtoolsParam` methods.

```
library(BiocParallel)

## Pi approximation
piApprox <- function(n) {
    nums <- matrix(runif(2 * n), ncol = 2)
    d <- sqrt(nums[, 1]^2 + nums[, 2]^2)
    4 * mean(d <= 1)
}

template <- system.file(
    package = "BiocParallel",
    "unitTests", "test_script", "test-sge-template.tmpl"
)
param <- BatchtoolsParam(workers=5, cluster="sge", template=template)

## start param
bpstart(param)
```

```
## Display param
param

## To show the registered backend
bpbackend(param)

## Register the param
register(param)

## Check the registered param
registered()

## Run parallel job
result <- bplapply(rep(10e5, 100), piApprox)

bpstop(param)
```

6 sessionInfo()

```
toLatex(sessionInfo())
```

- R version 4.1.0 (2021-05-18), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 20.04.2 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.13-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.13-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: BiocParallel 1.26.2
- Loaded via a namespace (and not attached): BiocManager 1.30.16, BiocStyle 2.20.2, R6 2.5.1, backports 1.2.1, base64url 1.4, batchtools 0.9.15, brew 1.0-6, checkmate 2.0.0, compiler 4.1.0, crayon 1.4.1, data.table 1.14.0, debugme 1.1.0, digest 0.6.27, ellipsis 0.3.2, evaluate 0.14, fansi 0.5.0, fs 1.5.0, highr 0.9, hms 1.1.0, htmltools 0.5.1.1, knitr 1.33, lifecycle 1.0.0, magrittr 2.0.1, parallel 4.1.0, pillar 1.6.2, pkgconfig 2.0.3, prettyunits 1.1.1, progress 1.2.2, rappdirs 0.3.3, rlang 0.4.11, rmarkdown 2.10, stringi 1.7.3, stringr 1.4.0, tibble 3.1.3, tools 4.1.0, utf8 1.2.2, vctrs 0.3.8, withr 2.4.2, xfun 0.25, yaml 2.2.1