batchtools configuration files

Configuration File

By default, batchtools looks for a configuration in the working directory with the name batchtools.conf.R
The simplest configuration file just defines the cluster functions. There are predefined cluster functions for all common schedulers. For example, to run batch jobs with slurm, the configuration file can look like this:

```
cluster.functions = makeClusterFunctionsSlurm("./simple.tmpl")
```

For other schedulers, the configuration files will looks similar. The predefined cluster functions are in the R directory, in files with names like clusterFunctionsInteractive.R, clusterFunctionsLSF.R, etc.

Brew Template

The cluster functions for most schedulers require brew templates. brew is a CRAN package which implements a templating framework.
Reference: https://cran.r-project.org/web/packages/brew/brew.pdf

Only two rules are necessary for cluster function templates. (1) R code between <% and %> is evaluated in place, and (2) R expressions between <%= and %> are evaluated and printed out.

BatchJobs templates vs. batchtools templates

In the BatchJobs github repository, there are sample templates in the examples directory. In the batchtools github repository there are, as of the writing of this document, no sample templates. Unfortunately, the BatchJobs samples can't be used for batchtools because of a change in the architecture of the job submission.

The cluster functions for a (scheduler or similar) are a set of functions that the framework calls to execute the jobs. All instantiations must define `submitJobs`, and most also define some or all of `listJobs`, `listJobsQueued`, `listJobsRunning`, and `killJob`.

Example:
Consider the function submitJob in the file clusterFunctionsSlurm.R in BatchJobs and batchtools. The signature of submitJobs in BatchJobs is:

```
submitJob = function(conf, reg, job.name, rscript, log.file, job.dir, resources, arrayjobs)
# etc.
```

It requires a brew template which will run the R script `rscript` and put the output in `log.file` and so forth.

The signature of submitJobs in batchtools is:

```
submitJob = function(reg, jc) {  # etc.
```

This requires a brew template which will extract the contents of the run script from the JobCollection object `jc`. A JobCollection is an R environment (i.e., a symbol table.) The names of the main objects in the environment can be found in the constructor in the file JobCollection.R and their usage must be deduced from the code.

- In an ordinary batch job execution run, there is one job per JobCollection.
- The character vector (string) `uri` stores the location of the job, in RDS.
- The job has a unique name derived from taking a hash of its contents. This hash is in the character vector `job.hash`.
- The character vector `log.file` is the location in the registry that the output should be stored.
- The character vector `file.dir` is the path to the registry.
- The character vector `job.name` is batchtools' name for the job.

In fact, in batchtools, the registry is an ordinary directory with sub-directories called jobs, logs, and results. The jobs are stored (in RDS) in the jobs directory, the output is in the logs directory and the results are in the results directory. (In BatchJobs, the registry used a database, but the database was replaced with R data.frames for batchtools.)

Slurm template

This is a sample slurm template, for running the EQTL problem.

```
#!/bin/bash
<%

    # make directory in registry
    rdir <- paste(file.dir, "/rexe", sep="")
    if (!dir.exists(rdir))  {
      dir.create(rdir, recursive = TRUE)
    }

    # make R script for job to run - rscript that runs JobCollection
    rsrc = paste(
          "library(batchtools)",
          "library(gQTLstats)",
          "library(GenomicRanges)",
          sprintf("jc = readRDS('%s')", uri),
          sprintf("batchtools:::doJobCollection(jc, output = '%s')", log.file),
          sep=";")

    rscript <- fp(file.dir, "rexe", sprintf("%s.R", job.hash))
    cat(rsrc,file=rscript,sep="\n")
```

```
    Sys.chmod(rscript, mode = "0777")

%>

#SBATCH --job-name=<%= job.name %>
#SBATCH --output=<%= log.file %>
#SBATCH --error=<%= log.file %>
#SBATCH -p compute
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -t 1:00

R CMD BATCH --no-save --no-restore "<%= rscript %>" /dev/stdout
```

Discussion:

1. This is significantly more abstruse than the analogous template for BatchJobs. I am not sure this is the simplest way to do this. I will check with Michel Lang to see if I have overlooked something obvious.
2. The general idea is to write out an R script which runs the batch job and to have the template run that script. It is almost possible to do without the script. To do without the script, the contents would be left in the variable rsrc and run with `Rscript -e "<%= rsrc %>"`. The difficulty is that one runs out of quotation marks quickly. The single-tick and double-quote are both in the thing to run, which leaves only backticks, and the command is easily confused no matter how you do it.
3. The upper half of the file is R code. This is evaluated when the submitJobs function instantiates the template. At this point, a tiny R script is written out for the job to run. The script is named with the job hash, to ensure uniqueness. The guts of the script load the job collection from the RDS and call `batchtools:::doJobCollection` to run it.
4. The lower half of the template is a standard slurm run script. It defines some parameters and runs the job.
5. An inconvenience is that the compute nodes must load any libraries the job requires, e.g., `gQTLstats`. This means the template has to be overhauled for each new use. There is probably a more elegant way to do this.
6. The slurm option -p is the name of the queue to run in. By default, cfncluster calls its slurm queue `compute`.

LSF template

For the sake of comparison, here is the corresponding LSF template. This one uses Rscript -e and does not write out an R file to the registry.

```
<%

    jt <- paste(file.dir, "/bout", sep="")
    if (!dir.exists(jt))  {
      dir.create(jt, recursive = TRUE)
    }
    jb <- fp(file.dir, "bout", sprintf("%s.bout", job.hash))
```

```
8
9     # rscript to run JobCollection
10    rscript = paste(
11            "library(batchtools)",
12            "library(GenomicRanges)",
13            "library(gQTLstats)",
14            sprintf("jc = readRDS('%s')", uri),
15            sprintf("batchtools:::doJobCollection(jc, output = '%s')", log.file),
16          sep=";")
17
18    queue = "short"
19
20 %>
21
22 #BSUB-o <%= jb %>
23 #BSUB-q short
24
25 Rscript -e "<%= rscript %>"
```

SGE template

The SGE template is more complex than the others because of the way SGE jobs are submitted.

```
1  <%
2
3     # make two additional directories in registry
4     rdir <- paste(file.dir, "/rexe", sep="")
5     if (!dir.exists(rdir))  {
6       dir.create(rdir, recursive = TRUE)
7     }
8
9     jdir <- paste(file.dir, "/exe", sep="")
10    if (!dir.exists(jdir))  {
11      dir.create(jdir, recursive = TRUE)
12    }
13
14    # make R script for job to run - rscript that runs JobCollection
15    rscript = paste(
16            "library(batchtools)",
17            "library(gQTLstats)",
18            "library(GenomicRanges)",
19            sprintf("jc = readRDS('%s')", uri),
20            sprintf("batchtools:::doJobCollection(jc, output = '%s')", log.file),
21          sep=";")
22
23    rf <- fp(file.dir, "rexe", sprintf("%s.R", job.hash))
24    cat(rscript,file=rf,sep="\n")
25    Sys.chmod(rf, mode = "0777")
26
27    # make bash script to run the R script
```

```
    jscript <- fp(file.dir, "exe", sprintf("%s.sh", job.hash))
    cat(paste0("/usr/local/bin/R CMD BATCH ", rf),file=jscript,sep="\n")
    Sys.chmod(jscript, mode = "0777")

%>

#$ -N <%= job.name %>
#$ -j y
#$ -S /bin/bash
#$ -V
#$ -o <%= log.file %>
#$ -l R=true
#$ -cwd

<%= jscript %>
```

Discussion:

- The SGE scheduler does not seem to have a way to include command line arguments or redirection on the command line. (Is this really true?) This template closely imitates the Channing cluster utility qbR, which creates a bash shell script to run the R script.
- The idea here is that submitJobs first writes a job submission script by using a brew template which creates an R script to run the job, then a bash script to run the R script. Then submitJobs runs this job submission script which runs the bash script which runs the R script which loads the real R script from RDS which does the work. All that can be said in favor of this is that it works.
- (There really must be a more elegant way to do this. I will ask Michel Lang what he recommends.)
- The line #$ -l R=true is necessary for the heterogenous Channing cluster. It requires a node with R installed to run the job. On cfnclusters it should be removed.

Summary

The simplest way to set up to run batchtools on a cluster is to put two files in the working directory: a configuration file named batchtools.conf.R and a brew template which is designed for the particular scheduler the cluster runs.