Simple batchtools example with slurm

Approximating the value of π

This is a description of how to run the piApprox example on a CfnCluster using slurm scheduler and batch tools. The piApprox example is described in `vignettes/batchtools.Rmd` in the batchtools repository.

I. Configuration file:

```
[aws]
aws_access_key_id = ### Your AWS Access Key ###
aws_secret_access_key = ### Your AWS Secret Access Key ###
aws_region_name = us-east-1

[cluster tiny]
vpc_settings = public
key_name = ### a private key from a pair AWS generated ###
base_os = centos6
initial_queue_size = 2
max_queue_size = 4
maintain_initial_size = false
master_instance_type = t2.micro
compute_instance_type = t2.micro
scheduler = slurm

[vpc public]
vpc_id = ### Your default VPC ###
master_subnet_id = ### Any subnet in your default VPC ###

[global]
sanity_check = true
update_check = true
cluster_template = tiny
```

The default location for the configuration file is `~/.cfncluster/config`, or the `--config` option can be used to indicate another file location. In cfncluster 1.3.2, slurm must be run on a CentOS6 system. (see note on the systemd bug.) To launch the cluster, execute:

```
$ cfncluster create tiny
```

Creation of the cluster usually takes about ten minutes. To watch the creation process, open the AWS CloudFormation console, click on the stack name and open the Events and/or Resources sections. When the

cluster is created, the instances will be visible in the EC2 console.

II. Files for the pi Approximation Example

The `/home` and `/shared` directories are NFS exported on the master node and mounted on the compute nodes. This means either of these is a good place to put a batchtools repository. The simplest place to run the tiny example is in `/home/centos`.

To run the example, we need to upload or create (vim is installed) three files. First, we need a one-line `batchtools.conf.R` file, like this:

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

Then, we need the brew template, `simple.tmpl`, like this:

```
#!/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)",
        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
```

Finally, as a convenience, an R file, `piApprox.R`, with the function in it.

```
piApprox = function(n)  {
```

```
  nums = matrix(runif(2 * n), ncol = 2)    # random point in [0,1] x [0,1]
  d = sqrt(nums[,1]^2 + nums[,2]^2)        # distance from origin to point
  4 * mean(d <= 1)                         # count points in unit circle
}
```

III. Install batchtools package

R-3.4.0 is installed on the cfncluster AMI's but we need to install batchtools, and (important) we want to install it in a place that all nodes can access. The `/shared` directory is a good place for this.

```
$ sudo mkdir -p /shared/R/Library
$ sudo chmod -R 777 /shared/R/Library
$ echo "R_LIBS=\"/shared/R/Library\"" >>.Renviron
$ R
> install.packages("batchtools", repos="http://cran.us.r-project.org")
```

IV. Run the example

At this point, it is a good idea to open another ssh session to the master node. This is so we can watch the jobs go to the queue in one window and type commands at the R prompt in the other. At the R prompt, run this:

```
> library(batchtools)
> source("piApprox.R")
> reg <- makeRegistry("./registry")
> n <- 10
> batchMap(fun = piApprox, n = rep(1e5,n))
> submitJobs()
> waitForJobs()                            # should be very fast
> reduceResults(function(x, y) x + y)/n    # should be about 3.141
```

After executing `submitJobs()` in the R session, execute `squeue` in the other session. This should show the queue and the nodes running the jobs. When the queue is empty, the `waitForJobs()` should return `TRUE`.

V. Autoscaling

The cluster will autoscale to four compute nodes (the maximum we allowed in the config file) if the conditions are right. We just need to put enough jobs in the queue to make the cluster monitor notice the extra work, launch instances, and bring them online before the queue empties.

```
> system("rm -Rf ./registry")
> reg <- makeRegistry("./registry")
> n <- 50                                  # more jobs
> batchMap(fun = piApprox, n = rep(1e6,n))  # bigger jobs
> submitJobs()
> waitForJobs()                            # should be very fast
> reduceResults(function(x, y) x + y)/n    # should be about 3.141
```

Note: Unfortunately, this demonstration does not work with slurm because slurm is too fast. All fifty jobs are done before the monitor notices the extra work. It works with SGE.

VI. Delete the cluster

To delete the cluster execute (on the local machine):

```
1 $ cfncluster delete tiny
```

References:

CfnCluster documentation: http://cfncluster.readthedocs.io/en/latest/index.html
CfnCluster main GitHub repository: https://github.com/awslabs/cfncluster