EQTL example with slurm

I. Bioconductor AMI

cfncluster 1.4 has been released, so custom AMI's need to be made again. The bug with ubuntu and slurm should be fixed.

Ubuntu 16.04 AMI: start with ami-6353ce19 (cfncluster ubuntu16.04 in us-east-1):

```
sudo apt-get update
sudo apt-get install -y r-base-core                    # brings in the R dependencies
sudo apt-get install -y libcurl4-openssl-dev           # devtools dependency
sudo apt-get install -y libssl-dev                     # devtools dependency
sudo apt-get install -y nfs-kernel-server              # NFS server (also EFS dependency)
sudo apt-get install -y libxml2-dev                    # XML dependency
sudo apt-get install -y default-jdk                    # R-3.4.3 dependency
sudo apt-get install -y libmariadb-client-lgpl-dev     # RMySQL dependency

# Install R-3.4.3 from source
wget https://cran.r-project.org/src/base/R-3/R-3.4.3.tar.gz
gunzip R-3.4.3.tar.gz
tar xvf R-3.4.3.tar
cd R-3.4.3
sudo ./configure --prefix=/usr/local/lib/R-3.4.3 --with-x=no
sudo make all
sudo make install

# cp /usr/local/lib/R-3.4.3/bin/R and Rscript to /usr/bin

sudo R      # install to system library
> install.packages("devtools", repos="http://lib.stat.cmu.edu/R/CRAN")
> install.packages("XML", repos="http://lib.stat.cmu.edu/R/CRAN")       # rtracklayer dependency

> source("https://bioconductor.org/biocLite.R")
> biocLite();
> biocLite("gQTLstats");   # ldblock and VariantAnnotation installed with this
> biocLite("geuvPack")
> install.packages("batchtools", repos="http://lib.stat.cmu.edu/R/CRAN")

# Remove R tarball and source from /home/ubuntu
```

Stop instance, make AMI. Result: ami-1d8dda67

II. Security Group

The cluster nodes will run in security group `bioc-efs-node-sg`. `bioc-efs-node-sg` allows all incoming traffic from members of its own group and SSH from anywhere. The inter-group traffic is necessary for cluster communications, and the SSH is so a user can log into the nodes.

III. Configuration File

```
[aws]
aws_access_key_id = # ACCESS KEY
aws_secret_access_key = # SECRET ACCESS KEY
aws_region_name = us-east-1

[cluster bioc]
vpc_settings = public
key_name = # KEY NAME (without .pem)
base_os = ubuntu1604
custom_ami = ami-1d8dda67
s3_read_resource = arn:aws:s3:::eqtl-example-bucket/*
post_install = s3://eqtl-example-bucket/postinstall.sh
initial_queue_size = 2
max_queue_size = 2
maintain_initial_size = false
master_instance_type = t2.small
compute_instance_type = t2.small
scheduler = slurm

[vpc public]
vpc_id = vpc-576d222f
master_subnet_id = subnet-66c6be02
vpc_security_group_id = sg-79b6b80c

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

Notes:

1. The `custom_ami` is a public Ubuntu 16.04 Bioconductor AMI.
2. The `vpc_security_group_id` is the ID of the group `bioc-efs-node-sg`.
3. The installed Bioconductor packages can't be run on a `t2.micro`
4. The options `initial_queue_size` and `max_queue_size` are misnomers. They are the initial and maximum number of nodes the cluster will launch.
5. Edit the file with credentials and a key, then run with: `cfncluster --config config.eqtl create bioc`

IV. Files for running the EQTL Example

The example can be run in `/home/ubuntu` or in `/shared`. Both are NFS shared among all nodes in the base Cfncluster AMIs. Files can be upoaded by sftp or edited at the command prompt (vim is installed.) Copies of these files are also in `/efs/btrun`.

1. `batchtools.conf.R`

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

2. `simple.tmpl`

```
#!/bin/bash

<%
# relative paths are not handled well by Slurm
log.file = normalizePath(log.file, winslash = "/", mustWork = FALSE)
-%>

#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

Rscript -e 'batchtools::doJobCollection("<%= uri %>")'
```

3. `eqtl.R`

```
library(gQTLstats)
library(ldblock)
library(VariantAnnotation)
library(geuvPack)
library(batchtools)

data(geuFPKM)
ss <- stack1kg()
v17 <- ss@files[[17]]
someGenes <- c("ORMDL3","GSDMB", "IKZF3", "MED24", "CSF3", "ERBB2",
               "GRB7", "MIEN1", "GSDMA", "THRA", "MSL1")
se17 <- geuFPKM[ which(rowData(geuFPKM)$gene_name %in% someGenes),]

n <- 10
vr0 <- 39.5e6
vr1 <- 40.5e6
v <- seq(vr0, vr1, length=n+1)
vl <- zipup(v[1:n],v[2:(n+1)]-1)

```

```r
run.job <- function(v, se, vcf)  {
  library(gQTLstats)
  library(ldblock)
  library(VariantAnnotation)
  library(geuvPack)
  vr <- GRanges("17", IRanges(start=v[1], end=v[2], names=c("range")))
  results <- AllAssoc(se, vcf, vr)
}

concat.job <- function(gr1, gr2)  {
  gr <- c(gr1,gr2)
}
```

V. Run EQTL Example

```r
R> library(batchtools)
R> source("eqtl.R")
R> system("rm -Rf ./registry")    # if there might be an old registry
R> reg <- makeRegistry(file.dir="./registry", conf.file = "./batchtools.conf.R")
R> batchMap(fun = run.job, as.list(vl), more.args = list(se=se17, vcf=v17))

R> submitJobs()
R> waitForJobs()    # could take about five minutes with only two nodes
R> all.results <- reduceResults(fun = concat.job)

R> save(all.results, file="all.results.RData")
```

The results can be downloaded by sftp to a local system and compared to locally generated results for correctness. Real output (as opposed to tiny debugging output) can also be saved to the EFS or to an S3 bucket for further processing.

VII. EFS Mount Target

Using the AMI for R computations will inevitably necessitate installing more R packages. Rebuilding the AMI every time new packages are installed is time-consuming. Installing in the cluster nodes' NFS will result in the packages being deleted with the cluster. An EBS volume can't be mounted by all cluster nodes simultaneously.

A good solution is to use an EFS file system. EFS is an AWS managed service that provides a shared, elastic, available and persistent file system. (Shared means all the nodes can mount it simultaneously. Elastic means it grows to accomodate the data stored in it. Available means there is no single server node (as in NFS) which can fail and bring down the whole system. Persistent means it is not deleted when the instances that mount it are deleted. It is also relatively affordable.)

Suppose the EFS is `fs-e9a39da0`, in `VPC-5763222f`. DNS name: `fs-e9a39da0.efs.us-east-1.amazonaws.com`

A. Security Group:

The EFS mount targets need a security group that permits incoming traffic from the cluster nodes. This group is called bioc-efs-mount-sg. bioc-efs-mount-sg allows incoming NFS traffic on port 2049 from bioc-efs-node-sg. The security group of the EFS mount targets can be set in the AWS EFS console.

B. Mount EFS:

The cluster nodes use a post-installation script to mount the EFS. The post-installation script is downloaded during cluster initialization from a S3 bucket. This requires two lines in the config file.
The `s3_read_resource` allows the cluster nodes access to the S3 bucket in which the postinstall script is located.

```
s3_read_resource = arn:aws:s3:::eqtl-example-bucket/*
post_install = s3://eqtl-example-bucket/postinstall.sh
```

The postinstall script just mount EFS and downloads `.Renviron` to the home directory

```
#!/bin/bash

sudo mkdir -p /efs
sudo mount -t nfs -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 fs-e9a39da0.efs.us-east-1.amazonaws.com:/ /efs

aws s3 cp s3://eqtl-example-bucket/dotRenviron /home/ubuntu/.Renviron
```

C. **Default library in `.Renviron`**

Installed packages go by default into EFS, so they do not need to be re-installed when a new cluster is created. Directory `/efs/R/Library` is owned by user ubuntu. The postinstall script downloads `.Renviron` to `/home/ubuntu` on the master node. This has the effect of setting it for all nodes, because `/home` is exported by NFS.

```
# .Renviron
R_LIBS="/efs/R/Library"
```

VIII. Next Steps

The next obvious step is to increase the size of the cluster and the size of the problem. The parallelism will become worthwhile only for reasonably large clusters and problems.