Cloud Computing with reactor

P. Pernot - 05/07/2018

Contents

1 Initial steps

Cf. also `https://openstack.lal.in2p3.fr/tutoriel/tutorial/`

1. Create an account in project "LCP" here
   `https://openstack.lal.in2p3.fr/demande-de-compte-cloudvd/`.
   The project administrator is "pascal.pernot".

2. Get file `https://openstack.lal.in2p3.fr/files/2016/02/terena.pem`
   and store it in `$HOME/.certs/terena.pem`

3. Create user configuration file for Cloud@VD `os_lal.rc` dans le répertoire `Script` du projet

```
export OS_USERNAME=***user***
export OS_PASSWORD=***pwd***
export OS_TENANT_NAME=LCP
export OS_PROJECT_NAME=LCP
export PS1='[\u@\h \W($OS_USERNAME @ OS_LAL)]\$ '
export OS_AUTH_URL=https://keystone.lal.in2p3.fr:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_CACERT=$HOME/.certs/terena.pem
export OS_USER_DOMAIN_NAME=u-psud
export OS_PROJECT_DOMAIN_NAME=u-psud
export OS_VOLUME_API_VERSION=2
```

   and source it

```
source Scripts/os_lal.rc
```

4. Install the OpenStack client from here (requires `python`)
   `https://pypi.org/project/python-openstackclient/`
   Test install with command

   ```
   openstack token issue
   ```

5. Create a keypair from you public key

   ```
   openstack keypair create --public-key $HOME/.ssh/id_rsa.pub myKey
   ```

   If you do not have public key, cf. `https://openstack.lal.in2p3.fr/tutoriel/tutorial/`

2 Configuration files

All files stored in directory `Scripts` of current `reactor` project.
Note: if you mod these files, there should be no space around '='.

2.1 `cl_config.rc`

Computing configuration (nb. runs, nb VMs...). The parameters and constraints are:

- `CL_SIZE` : number of VMs (max=8/9; I keep 1/2 for other projects)
- `NB_CORE` : number of cores on each VM.
  The larger, the more difficult to create new instances of VM (keep below 16).
  There is a hard constraint `CL_SIZE*NB_CORE` $\leq$ `200`.
- `RUN_BY_CORE` : integer chosen such as `MC_RUNS=RUN_BY_CORE*CL_SIZE*NB_CORE` is as close as possible to desired nb. of runs (*ex.* 6 or 7 for 500 runs [6/7]*8*10=[480/540]).
  The idea is to run in parallel `NB_CORE` MC loops of `RUN_BY_CORE` steps on each VM. Each run has a unique tag/number, and all process store the final results in the VM's `MC_Output` directory.

```
export CL_SIZE=8      # Size of cluster
export NB_CORE=10     # Nb cores on VM

RUN_BY_CORE=5
# Nb runs, ensures divisibility by CL_SIZE and NB_CORE
export MC_RUNS=$((RUN_BY_CORE*CL_SIZE*NB_CORE))

export VM_FLAVOR=os.$NB_CORE
export VM_IMAGE=titan_2018-07-04  # Image to be run (Debian + Fortran)
export VM_KEY=***myKey***         # Name of OpenStack keyPair
```

3 Running the code

Set of commands to start, monitor and terminate jobs.

Scripts/claunch.sh Create `titan_x (x in 1..CL_SIZE)` VMs as defined in `Scripts/cl_config.rc`

Scripts/crun.sh Copy datasets and start reactor jobs on VMs. Each core has a defined range of tags to process.

Scripts/cmonitor.sh Shows the existing results files on all VMs

Scripts/cgather.sh Collects all results files and store them in local `MC_Output.`

Scripts/cclean.sh Remove all files on VMs. Useful only if VMs are to be used for other task(s).

Scripts/cdel.sh Destroy VMs.

3.1 Typical session

```
source Scripts/os_lal.rc
Scripts/claunch.sh
Scripts/crun.sh
... Wait ...
Scripts/cmonitor.sh
... Wait till all jobs done...
Scripts/cgather.sh
Scripts/cdel.sh
```

4 Have fun !