How to use Skyway

Igor Yakushin

April 3, 2021

1 What is Skyway

- Skyway provides a way to submit a job to the Google Cloud or other clouds using the same Slurm scheduler as on midway cluster at RCC.
  - You can submit a job in batch
  - You can use jupyter interactively
- At the moment MSCA program is using Skyway to submit jobs to GCP only
- Skyway is created and administered by RCC, in particular Yuxing Peng.

2 Why to use Skyway

- Skyway provides an easy way for MSCA program to connect all the students to the same financial account that pays for your Google Cloud usage
- It also provides a control for RCC sysadmins to prevent you from overspending which is so easy to do when you are using GCP directly:
  - You are given some budget
  - Skyway would not allow you to submit a job that would go over your budget
  - If you are using Skyway in a reasonable way, for school purposes only, your budget can be extended, the exact procedures for that to be determined later. For now, if you run out of your allocation, contact me and Albert Chan to extend it.
- Skyway provides a way for RCC sysadmins to control who can use/share what resources, for example, storage.
- There seems to be no straightforward way to accomplish all the above by using Google Cloud directly
- If you use Google Cloud directly, you are on your own:

  - you would either have to use your trial credits or pay there, the school would not pay for your cloud usage
  - you can easily overspend by simply forgetting to shut down some resources

3 What resources are available on Skyway

Currently there are the following groups of nodes available in msca-gcp partition, distinguished by Slurm constraint:

- 16 preemptible nodes with A1 GPU, about 70G of RAM, 6 CPU cores, slightly over $1 per hour.
  - This should be your first choice for Deep Learning kind of jobs that require GPU.
  - To use it, set `--constraint=a1` in your batch script or sinteractive.
- 20 non-preemptible node with V100 GPU, about 70G of RAM, 6 CPU cores, slightly over $3 per hour
  - Can be used for Deep Learning but 3 times more expensive than a1, previous generation of GPU
  - It's only advantage over a1 is that it is non-preemptible.
  - Avoid using it unless you absolutely cannot be interrupted or need to run for more than 24 hours.
  - `--constraint=v1`
- 10 preemptible nodes with 30 CPU cores, about 200G of RAM, no GPU, about $0.7 per hour.
  - This is good for big data kind of jobs to use with Spark
  - `--constraint=c30p`
- 4 small 1 CPU nodes with 16G of RAM, $0.05 per hour.
  - This might be suitable for software installation or for experimenting with the system
  - `--constraint=c1`

The kinds of available nodes might be rapidly changing depending on demand. To find out what groups of nodes are currently availble and what kind of nodes each group has, run

```
sinfo | grep msca
```

Currently, the output from it, shows (after some formatting to make it more readable):

```
msca-gcp-a1-[001-016],
msca-gcp-c1-[001-004],
msca-gcp-c30p-[001-010],
msca-gcp-v1-[001-020]
```

To find out what resources are on a particular node, run

```
scontrol show node NODENAME
```

where you can get some NODENAME from the group of interest from the previous command, for example, msca-gcp-a1-001 can be used as a representative for a1 group. All the nodes in the same constraint should be identical, so select any to query for resources.

If necessary, we can add any kind of nodes that are available on GCP directly.

'Preemptible' above means that you cannot run on such a node for more than 24 hours and even within this period your job theoretically might be interrupted. I have not yet seen interruptions and not sure how common they are but just in case you need to checkpoint periodically to be able to restart from the latest checkpoint rather than from scratch. 'Preemptible' nodes a1 and c30p are chosen for costs reason. Non-preemptible nodes are much more expensive: compare for example v1 and a1: a1 has the current generation of Nvidia GPU card - A100 - while v1 has the previous generation - V100. Yet preemptible a1 is 3 times cheaper than non-preemptible v1.

4 How to use Skyway

- Skyway login node is only visible on the internal UChicago network. So either use VPN or ssh from midway login node
- `ssh YourUserName@skyway.rcc.uchicago.edu`
- The only file system that skyway mounts from midway, is `/project2/msca/$USER`, use it to exchange data with midway or do it with scp or rsync.
- Your disk space on the cloud is mounted at `/cloud/msca-gcp/$USER` both on skyway login node and GCP compute node, use it to exchange data with the cloud
- After login, `cd /cloud/msca-gcp/$USER` because some commands, like sinteractive might not work from `$HOME`
- Use usual Slurm commands to submit a batch job to a node in the cloud.
- Here is an example batch script, `test1.batch`:

  ```
  #!/usr/bin/bash
  #SBATCH --account=msca-gcp
  #SBATCH --partition=msca-gcp
  ```

```
#SBATCH --nodes=1
#SBATCH --exclusive
#SBATCH --constraint=a1
#SBATCH --gres=gpu:1
#SBATCH --time=3:00:00

# set up the environment for python
source /software-msca/etc/env1.sh
which python

# go to the directory with your code
cd /cloud/msca-gcp/$USER/project1/code
pwd

# train the model
echo "Training"
date
python train_unet.py

# predict
echo "Predicting"
date
python predict.py
date
```

- Submit it in batch as usual: `sbatch test1.batch`
- If you want to use different kind of node, change constraint.
- The conda environment that is sourced was used at Advanced Machine Learning class in Winter 2021 semester
- You can create your own or use what is provided. Take a look inside `/software-msca/etc/env1.sh`.
- When you submit a job, it takes about a couple minutes to start GCP node. When your batch job finishes, the node shuts down automatically.
- If you want to use the node interactively with jupyter, submit the following batch job:

```
#!/usr/bin/bash
#SBATCH --account=msca-gcp
#SBATCH --partition=msca-gcp
#SBATCH --nodes=1
#SBATCH --exclusive
#SBATCH --constraint=a1
#SBATCH --gres=gpu:1
```

```
#SBATCH --time=3:00:00

cd /cloud/msca-gcp/$USER
HOST=‘hostname‘
IP=‘grep $HOST /skyway/files/etc/hosts | awk ’{print $1}’‘
echo "Run jupyter-notebook at $HOST/$IP" > notebook.log
TOKEN=‘openssl rand -base64 18‘
echo "URL AT http://${IP}:8888/?token=${TOKEN}" >> notebook.log
module load anaconda3
conda activate /software-msca/conda_envs/AML
jupyter-notebook --ip=0.0.0.0 --NotebookApp.token=${TOKEN} \
                             --no-browser > .notebook.log 2>&1
```

- After your job starts running in couple minutes - check with squeue - the URL to copy and paste into your browser will be recorded into `notebook.log` in the current directory
- Once you are in jupyter, select AML kernel if you want to use the provided environment.
- If you create your own environment on your account, make sure to install there `nb_conda` to see the corresponding kernel in jupyter. I would also install conda, pip, jupyter.
- Once you finish working with the notebook, kill the corresponding job with `scancel`, otherwise, it will run for the time you specified.
- To create your own environment, use `sinteractive` to log into the node
- When submitting a job, your estimated cost/balance info is printed. If the requested time would overspend your budget, the job is rejected.
- You can also query your actual (but somewhat delayed) balance by executing: `usages`
- If you have problems, submit a ticket to RCC and cc me and Yuxing by sending e-mail to: `help@rcc.uchicago.edu`, `ivy2@uchicago.edu`, `yuxing@uchicago.edu`
- Skyway at the moment is rather experimental.
- The two batch scripts shown above can be found on skyway in `/software-msca/examples/`. Copy and modify for your needs.