Sbox: simple toolbox for Slurm

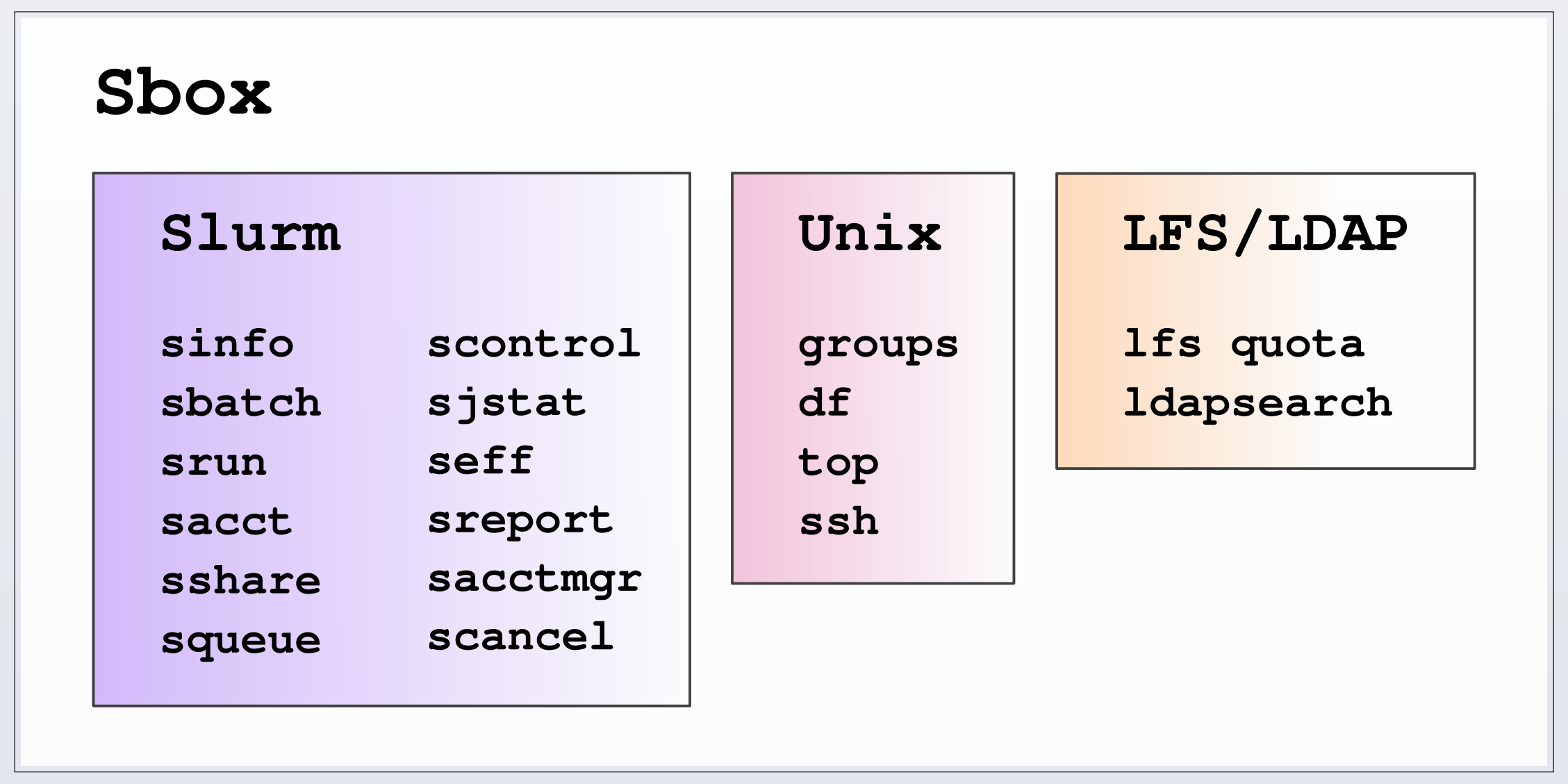
Ashkan Mirzaee

Research Computing Support Services
University of Missouri

University of Missouri

Introduction

Slurm is one of the most popular workload managers among HPC clusters. Slurm provides numerous commands and options for resource allocations and monitoring activities. Applying the large number of the commands and options, can be very challenging for the new cluster users. Sbox is a simple and lightweight Python toolbox for Slurm that collected a set of of Slurm and Unix commands at one place.



Sbox

Slurm

sinfo scontrol
sbatch sjstat
srun seff
sacct sreport
sshare sacctmgr
squeue scancel

Unix

groups
df
top
ssh

LFS/LDAP

lfs quota
ldapsearch

Figure 1:Slurm and Unix commands that are applied in Sbox.

Sbox is designed to provide important information about users' activities and cluster resources, as well as facilitate resource allocations on a HPC cluster. Sbox includes two commands: **`sbox`** and **`interactive`**.

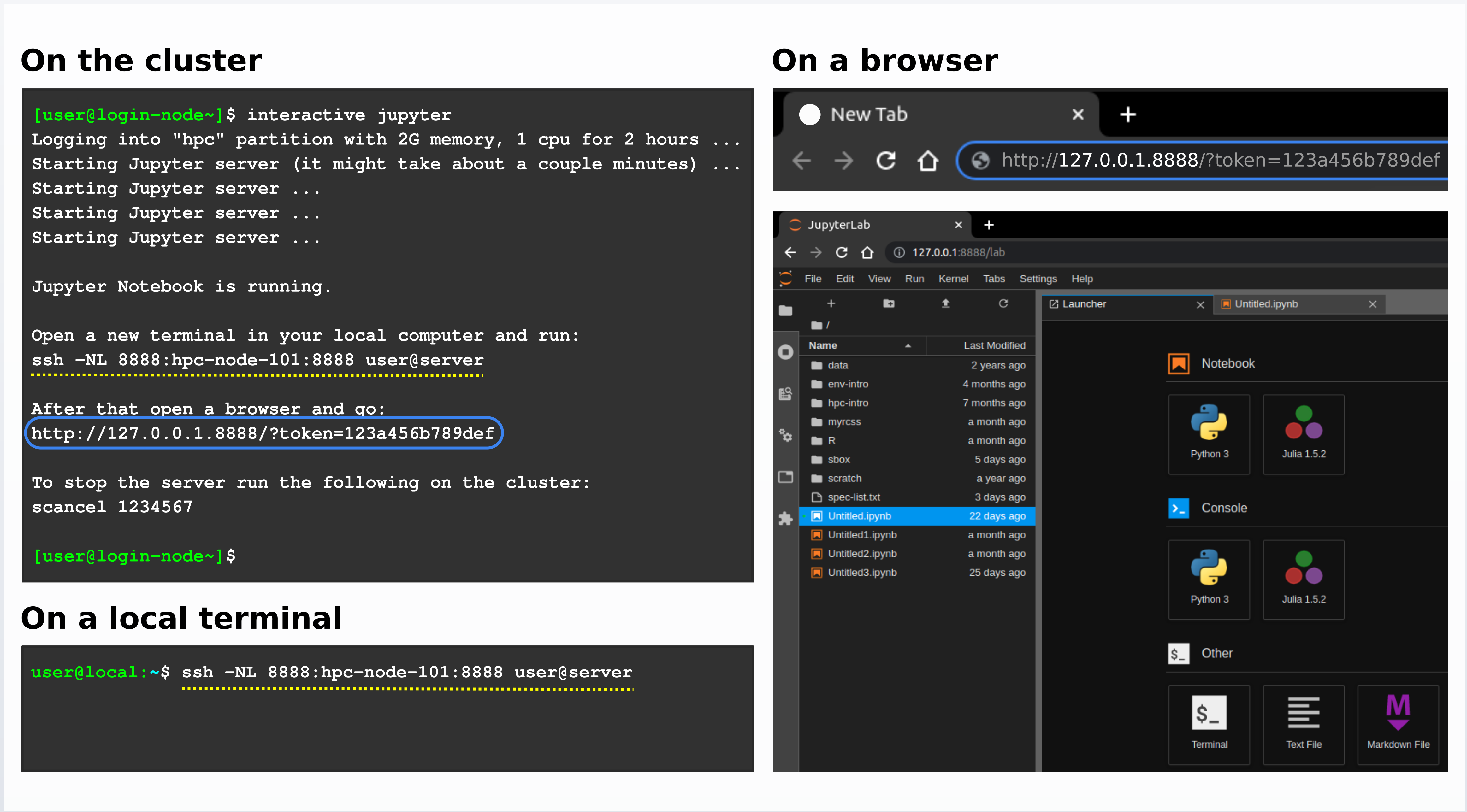
sbox

`sbox` includes various Slurm commands at one place and help users find the information about their activities and cluster resources. Beyond the Slurm commands, **`sbox`** provides some Unix features including users' groups, disk quotas or starting ssh agents.

sbox -h

```
-h, --help                       show this help message and exit
-a, --account                    show slurm accounts
-f, --fairshare                  show fairshare
-g, --group                      show posix groups
-q, --queue                      show jobs in the queue
-j JOBID, --job JOBID            show a running/pending job info
-c, --cpu                        show computational resources
-p, --partition                  show partitions
-u UID, --user UID               user id
-v, --version                    show program's version number and exit
--eff JOBID                      show efficiency of a job
--history {day,week,month,year}  show jobs history for last day/week/month/year
--pending                        show pending jobs
--running                        show running jobs
--qos                            show quality of services
--quota                          show quotas
--ncpu                           show number of available cpus
--ngpu                           show number of available gpus
--gpu                            show gpu resources
--license                        show available licenses
--reserve                        show reservation
--topusage                       show top usage users
--whodat UID                     show users informations by uid
--whodat2 UNAME                  show users informations by name
--agent {start,stop,list}        start/stop/list ssh-agents on the current host
```

Table 1:`sbox` command line options.



On the cluster

```
[user@login-node~]$ interactive jupyter
Logging into "hpc" partition with 2G memory, 1 cpu for 2 hours ...
Starting Jupyter server (it might take about a couple minutes) ...
Starting Jupyter server ...
Starting Jupyter server ...
Starting Jupyter server ...

Jupyter Notebook is running.

Open a new terminal in your local computer and run:
ssh -NL 8888:hpc-node-101:8888 user@server

After that open a browser and go:
http://127.0.0.1.8888/?token=123a456b789def

To stop the server run the following on the cluster:
scancel 1234567

[user@login-node~]$
```

On a local terminal

```
user@local:~$ ssh -NL 8888:hpc-node-101:8888 user@server
```

On a browser

Name	Last Modified
data	2 years ago
env-intro	4 months ago
hpc-intro	7 months ago
myrcss	a month ago
R	a month ago
sbox	5 days ago
scratch	a year ago
spec-list.txt	3 days ago
Untitled.ipynb	22 days ago
Untitled1.ipynb	a month ago
Untitled2.ipynb	a month ago
Untitled3.ipynb	25 days ago

Figure 2:The `interactive jupyter` command starts a JupyterLab session on a computational node and shows how to join the session.

interactive

`interactive` is an alias for using cluster interactively using Slurm **`srun`** and **`sbatch`** commands. The **`interactive jupyter`** provides a JupyterLab interface for using scientific software including Python, R, Julia, and their libraries. The command submits a batch file to start a Jupyter server on the cluster. Multiple kernels and environments can be applied to use different software and packages in JupyterLab.

interactive [jupyter] -h

```
-h, --help            show this help message and exit
-a, --account         slurm account name or project id
-n, --ntasks          number of tasks (cpus)
-N, --nodes           number of nodes
-p, --partition       partition name
-t, --time            number of hours (up to 8)
-k, --kernel          Jupyter kernel for python, r, julia
-e, --environment     virtual environment(s) for a JupyterLab session
-E, --myenv           path to a local virtual environment
-l, --license         license
-m, --mem             amount of memory (per gb)
-g, --gpu             number of gpus
```

Table 2:`interactive` command line options.

The **`interactive jupyter`** command uses Anaconda for running Python, R, and many scientific packages.

Features

- Access to many Slurm features at one place.
- Facilitate request resources interactively.
- Easy ability to start a JupyerLab session.
- JupyterLab interface with multiple kernels.
- JupyterLab interface with access to premade virtual environments such as TensorFlow and PyTorch.
- JupyterLab interface with access to a local virtual environments.
- Easy to set up and configure. It can be installed in the user level or cluster-wide.
- Explanatory help options (**`--help`**) and reference manuals (**`man sbox`**, **`man interactive`**).
- Improving **`seff`** command by using **`top`** command for showing the running jobs efficiency.
- Managing users ssh-agent to be able to communicate with clients outside (e.g. GitHub) or within the cluster without asking for the passphrase (users need the passphrase to start the ssh-agent).
- Helping users by showing their fairshares, accounts, quotas, jobs' history, running and pending jobs, as well as cluster resources.

Quick install

- Download and extract the latest Sbox release.
- Install Anaconda and create the required virtual environments and modulefiles.
- Update the **`config`** file based on the cluster information.
- Place a modulefile for Sbox under **`$MODULEPATH/sbox`** and load the module or add the Sbox bin directory to **`$PATH`**.

config

```
{
    "disk_quota_paths": [],
    "cpu_partition": [],
    "gpu_partition": [],
    "interactive_partition_timelimit": {},
    "jupyter_partition_timelimit": {},
    "partition_qos": {},
    "kernel_module": {},
    "env_module": {}
}
```

Table 3:Sbox configuration file.

Requirements

Sbox requires Slurm and Python >= 3.6.8. The **`interactive jupyter`** command requires Anaconda and an environment module system (e.g. Lmod) in addition to Slurm and Python. To use R and Julia from a JupyterLab session, we need R and irkernel as well as Julia to be installed. Review Sbox docs for installing Anaconda and creating the required virtual envs and modulefiles. Note that **`sbox`** options require some other commands. Review the options requirement in here.

Acknowledgements

This work would not have been possible without support and guidance of my colleagues at the University of Missouri-Columbia Research Computing Support and Services. Special thanks to Asif Magdoom, Predrag Lazic, Brian Marxkors, and Philip Redmon for their insights and progressive comments and reviews.

Sbox

- Author: Ashkan Mirzaee
- Source: github.com/ashki23/sbox
- Documentation: sbox.readthedocs.io