How to use Singularity on Lsens2

Stefan Lang

May 2021

Please read this!

When the blades are running a lot of scripts I normally (!!) loose access to the labbook hosted on a blade. I assume that is due to an overload on the internal network system that connects the blades with the frontend and the NAS systems. If e.g. 10 star processes try to save there outfiles to the NAS at the same time this not only slows down the star processes themselves, but renders the whole blade system unusable for anybody else.

So PLEASE use the $SNIC_TMP variable as much as possible. As star temp folder, as samtools tmp folder - yes why not as outfile / prefix folder? You can copy all outfiles you really need after the main script has finished. Much like the last step in the Python analysis I told you about. This would not only speed your scripts up, but would help everybody else, too!

1 Introduction

The aurora system as likely any other hpc system consists of login computers (frontends), storage units (NAS) and calculation nodes (blades). When you log into the system your session occupies RAM space and computing power from the frontend. Having multiple people logged into the system easily creates a lot of traffic there. So all heavy calculation should be performed on the blades.

But how do you work on your results from the heavy jobs on aurora?

I have to admit, that I most of the time have been working on the frontend. Many other have run the analysis on the frontend, too, but with the high cell count single cell experiments getting more common now the frontend is too small for all of us!

As this is not our only problem on aurora-ls2 I have created Singularity images and documented that on my github page. I frequently use the Makefile and the shell.sh.

But all of this is not important for you to use the Singularity images on aurora.

2 Use installed Singularity Images on Aurora-ls2

To use the modules you first need to add a module path to your .bashrc:

```
echo "module use /projects/fs1/common/modules/" >> ~/.bashrc
```

To load this in your active terminal you need to:

```
source ~/.bashrc
```

And finally you are ready to start the Singularity session on a blade. I have several sbatch scripts that start the image on a blade in different configurations:

```
/projects/fs1/common/software/SingSingCell/1.1/runSbatch_n10_t24h.sh
/projects/fs1/common/software/SingSingCell/1.1/runSbatch_n20_t24h.sh
/projects/fs1/common/software/SingSingCell/1.1/runSbatch_n1_t24h.sh
/projects/fs1/common/software/SingSingCell/1.1/runSbatch_n40_t12h.sh
/projects/fs1/common/software/SingSingCell/1.1/runSbatch_n20_t12h.sh
/projects/fs1/common/software/SingSingCell/1.1/runSbatch.sh
```

I assume you get the differences.
The contents of runSbatch_n1_t24h.sh:

```
#! /bin/bash
#SBATCH -n 1
#SBATCH -N 1
#SBATCH -t 24:00:00
#SBATCH -A lsens2018-3-3
#SBATCH -p dell
#SBATCH -J SingSing
#SBATCH -o SingSing.%j.out
#SBATCH -e SingSing.%j.err
module purge
module load Singularity/default SingSingCell/1.1
exit 0
```

Start the session like this:

```
sbatch /projects/fs1/common/software/SingSingCell/1.1/runSbatch_n1_t24h.sh
```

This will give you a typical answer (your process id will likely be different):

```
Submitted batch job 779229
```

Created the outfiles SingSing.779229.err and SingSing.779229.out. The .err file will tell you how to connect to the server:

```
.
.
.
    To access the server, open this file in a browser:
        file:///home/stefanl/.local/share/jupyter/runtime/jpserver-196980-open.html
    Or copy and paste one of these URLs:
        http://ls2-n3:9734/lab?token=92021f109e2d763a3a19af92dee13a815cfe7f5c7885a1e2
     or http://127.0.0.1:9734/lab?token=92021f109e2d763a3a19af92dee13a815cfe7f5c7885a1e2
```

From these links you can open the second one and get a working Jupyter notebook instance (in firefox).

3 Working with large files on the node

This whole document is about single cell experiments. I have had occasions where a Scanpy analysis of less than 40.000 cells resulted in a 100Gb(!) anndata file. This file would not load over the network. You remember the hpc setup from the introduction?
All our data is stored on the NAS. Therefore all computing systems need to copy or access data over the network. But the blades have local storage, too: The environment variable $SNIC_TMP. You need to copy your data before you do anything else to your working blade:

A typical Python analysis for me nowadays would look like this:

```
# I load all the libs
...
# I define input files
ifile = "someInfile.h5ad"

if 'origWD' not in locals():
    origWD = os.getcwd()

print(f"origWD = '{origWD}'" )

ofile=f"someOutfile.h5ad"
```

After this I normally copy the infile to the node and start working:

```
from shutil import copyfile

print(os.environ['SNIC_TMP'])
```

```
path = os.path.basename( os.environ['SNIC_TMP'] )
wd = f"/mnt/{path}/"
print ( f"wd = '{wd}'" )
ensure_dir( wd )

## Copy the infile to the blade: ##
print( f"cp {origWD}/{ifile} {wd}" )
target = f"{wd}/{ifile}"
if not os.path.isfile(target):
    copyfile( ifile, target)
#print( f"cp {origWD}/{ifile2} {wd}" )
#target = f"{wd}/{ifile2}"
#if not os.path.isfile(target):
#    copyfile( ifile2, target)
###################################

os.chdir( wd )

with open( 'Subset_And_Stats.versions.txt', 'w' ) as f:
    scanpy.logging.print_versions( file=f )
```

Then you can go on as if you would be working on the frontend. The labbook will still be saved directly on the NAS, but all outfiles you need to copy to the NAS later on:

```
origOutpath = f"{origWD}/analysisSession"
if os.path.exists(origOutpath):
    print( "orig outpath exists")
else:
    print("orig outpath does not exixst")
shutil.copytree( wd, origOutpath )
```

The end needs some tweaking. But it gets the job done.

I hope this document can be of help for you!

4 Final remarks

When the blades are running a lot of scripts I normally (!!) loose access to the labbook hosted on a blade. I assume that is due to an overload on the internal network that connects the blades with the frontend and the NAS systems. If e.g. 10 star processes try to save there outfiles to the NAS at the same time this not only slows down the start processes themselves, but renders the whole blade system unusable for anybody else.

So PLEASE use the $SNIC_TMP variable as much as possible. As star temp folder, as samtools tmp folder - yes why not as outfile / prefix folder? You can copy all outfiles you really need after the main script has finished. Much like

the last step in the Python analysis I told you about. This would not only speed your scripts up, but would help everybody else, too!