# Using SSD Disk Space

## Introduction

We have added SSDs, solid state disks, on a few compute nodes:

- These are local fast disks that can speed up applications that preform a lot of intensive I/O operations,
  - hence such jobs should complete faster when using SSDs.
- Jobs that do not perform intensive I/Os should not use the SSDs - this is a scarce shared resource.

- Since these disks are local to the compute nodes:
  - you cannot see the SSD from either login nodes,
  - your job will be able to use the SSD *only while the job is running*, hence
    - you need to prepare the files needed for a job prior to submitting the job, using /pool or /scratch, and
    - request the right amount of SSD disk space:
      - this is the maximum amount of disk space your job will need on the SSD at run-time, (similarly to the maximum of memory it will need).
  - Your job script will have to
    - copy those files to the SSD before processing them,
    - be adjusted to use the SSD,
    - upon completion, copy the results from the SSD elsewhere (/pool or /scratch), and
    - delete what you wrote on the SSD.
  - If you exceed the amount of disk space, i.e., your quota, your job won't be able to write any longer to the SSD, hence your job script should stop when encountering such error.

⚠️ Since we have only a few of them, they should be used only if your application greatly benefit from using an SSD.

The SSDs are currently only available for the `uTSSD.rq` queue, a restricted queue. Contact Sylvain or Rebecca if you want to be authorized to use it.

## HowTo

Since you can't access the SSDs from a login node, you must prepare the data the job will need somewhere else, like on `/pool` (or on `/scratch`) before submitting a job.

Like for memory, you need to guestimate how much SSD space your job will need. You will not be able to use more SSD space than you requested.

Remember, your job will still be able to access the `/home, /data, /pool,` and `/scratch` disks, hence you don't have to copy everything on the SSD,

only the I/O intensive part of the analysis should use the SSD.

### How to Prepare my Data

1. Create a subdirectory in `/pool` (or `/scratch`) and move or copy the data you will need, for example (as user smart1)
   ```
   cd /pool/genomics/smart1
   mkdir -p great/project/wild-cat
   ```
   Now is have a directory for this case, and would copy the I/O intensive part of the required data set in it.

2. While not required, you can pack these data in a compressed tar-ball
   ```
   cd /pool/genomics/smart1/great/project/wild-cat
   tar cfz ../wild-cat.tgz .
   ```
   The file `/pool/genomics/smart1/great/project/wild-cat.tgz` now holds you input data set,
   being compressed it is likely to be smaller than the content of `/pool/genomics/smart1/great/project/wild-cat,`
   That directory can be deleted, unless you will need it later.

3. Ancillary data and/or configuration files that are not causing intensive I/O can stay on a location under `/pool` (or `/scratch`)

### How to Adjust a Job Script to Use the SSD

Your jobs script will need the following 4 parts

#### Part 1: Copy the Data to the SSD

- At the top of your job script, load the `tools/ssd` module and copy or extract your data set as follows:

**example using recursive copy**

```
module load tools/ssd
cp -pR /pool/genomics/smart1/great/project/wild-cat/* $SSD_DIR/.
```

or

**example using compressed tar-ball**

```
module load tools/ssd
cd $SSD_DIR
tar xf /pool/genomics/smart1/great/project/wild-cat.tgz
```

The advantage of the compressed tar-ball is that the `.tgz` file is likely to be smaller than the content of the directory, hence less I/O transfer from the `/pool` disk, while un-compressing and writing to the SSD is fast,

## Part 2: Adjust the Script or a Configuration File

- You need to replace all references to `/pool/genomics/smart1/great/project/wild-cat` by `$SSD_DIR`,
- this can be easily done at the shell script level, but not in a configuration file, i.e., for flags/options
  `execute -o /pool/genomics/smart1/great/project/wild-cat/result.dat`
  is replaced by
  `execute -o $SSD_DIR/result.dat`
- Here is a simple trick to modify a configuration file:
  - Let's assume that your analysis uses a file `wow.conf`, where for instance the full path of some files must be listed, like:

    **wow.conf**

    ```
    # this is the configuration file of the fabulous WOW package
    input=/pool/genomics/smart1/great/project/wild-cat/wiskers.dat
    output=/pool/genomics/smart1/great/project/wild-cat/tail.dat
    paws=4
    eyes=2
    ```

  - Replace the `wow.conf` file by a `wow.gen` file as follows:

    **wow.gen**

    ```
    # this is the configuration file of the fabulous WOW package
    input=XXXX/wiskers.dat
    output=XXXX/wild-cat/tail.dat
    paws=4
    eyes=2
    ```

  - And create the `wow.conf` file from the `wow.gen` at run-time by adding the following in the job script:

    ```
    sed "s=XXXX=$SSD_DIR=" wow.gen > wow.conf
    ```

    As long as `XXXX` is not used for anything else, this will replace every occurrence of `XXXX` by the value of the environment variable `SSD_DIR`.

## Part 3: Run the Analysis

- With your data copied to the SSD and with your commands and configuration files adjusted to use the SSD, run your analysis.

## Part 4: Copy the Results from the SSD

- At the end of the job script, you must add instructions to copy the results of your analysis back to `/pool`(or `/scratch`, or `/data`).

  If/when the results are easily identifiable, you can use `the commands mv or tar,` and `find`, here are a few examples:

1. Move the directory where all the results are stored and the log file, delete the rest.

**moving identifiable results, delete the rest**

```
# move results and log file back
cd $SSD_DIR
mv results /pool/genomics/smart1/great/project/wild-cat/.
mv wow.log /pool/genomics/smart1/great/project/wild-cat/.
#
# delete the rest
rm -rf *
```

2. Move the directory where all the results are stored and the log file, delete the input (conservative approach, in case you missed something).

**moving identifiable results, delete known input sets**

```
# move results and log file back
cd $SSD_DIR
mv results /pool/genomics/smart1/great/project/wild-cat/.
mv wow.log /pool/genomics/smart1/great/project/wild-cat/.
#
# delete input set and other stuff
rm -rf input
rm wow.gen wow.conf
```

3. Move using the `--update` flag of `mv` (see `man mv`)

**moving using --update**

```
# move results using --update
cd $SSD_DIR
mv --update * /pool/genomics/smart1/great/project/wild-cat/.
#
# delete the rest
rm -rf *
```

Note, you can use `mv --update` on an explicit list (of files, directories, or file specification), not just * (everything), and you do not have to remove the rest, but can only remove what you know you can safely remove (conservative approach).

4. Find newer files and move them: the trick is to create a 'timestamp' file *before* starting the analysis.
That file can be used later to find any newer file with the `--newer=` option of `tar` (see `man tar`):

**Using a timestamp file and tar --newer=**

```
# set the timestamp
date > $SSD_DIR/started.txt
# run the analysis
...
# copy the new files in the subdir data/ to a compressed tar-ball
cd $SSD_DIR
tar --newer=$SSD_DIR/started.txt -cfz /pool/genomics/smart1/great/project/wild-cat-results.tgz
data/
# now remove it
rm -rf data/
# etc...
# delete everything, unless
rm -rf *
```

See previous comments and what to tar and what to remove: once you've `tar`'d new stuff in `data/`, remove `data/`, etc.

5. Using the timestamp file and the `find` command (see `man find`):

**Using find and a timestamp file**

```
# set the timestamp
date > $SSD_DIR/started.txt
# run the analysis
...
# find the new files in the subdir data/
cd $SSD_DIR
find data/ -newer $SSD_DIR/started.txt -type f > /tmp/list
# do the same on logs/, append to the list
find logs/ -newer $SSD_DIR/started.txt -type f >> /tmp/list
# etc...
# now save what is in the list with one tar
tar --files-from=/tmp/list -cfz /pool/genomics/smart1/great/project/wild-cat-results.tgz data/
# now remove data/ and logs/
rm -rf data/ logs/
# etc...
# delete everything, unless
rm -rf *
```

There are many more ways to accomplish this ....

BTW, the advantage of writing a `.tgz` file, rather than moving files is two fold, assuming your stuff is compressible:

a. You write less in the .tgz file, so it should be done faster (reading and compressing should be fast, writing is the slow step)
b. you need less disk space for your output (since it is compressed).

The drawback being that you need to know how to handle/view/deal with a `.tgz` file.

## A Pseudo Example

Here is what a job script might look like:

**Pseudo Example**

```
#
#$ -N example
#$ -o example.log -cwd -j y
#$ -q uTxlM.rq
#$ -l ssduse=2560G,hm,mres=20G,h_data=20G,h_vmem=20G
#
# pseudo example using a fake package WOW, on the SSD
#
echo $JOB_NAME started `date` on $HOSTNAME in $QUEUE jobID=$JOB_ID
#
module load tools/ssd
module load special/wow
#
# create a wow config file from a generic version, to insert the SSD temp dir value
sed "s=XXXX=$SSD_DIR=" ~/wow/wild-cat.gen > ~/wow/wild-cat.conf
#
# cd to the SSD temp dir and copy the data set to it, using the existing .tgz file
cd $SSD_DIR
tar xf /pool/genomics/smart1/great/project/wild-cat.tgz
#
# create some sub dirs for output and logs
mkdir output
mkdir logs
#
# run the wow analysis (note how some files are not on the SSD)
wow --type=m --params=$HOME/wow/parameters.dat --config=$HOME/wow/wild-cat.conf -o $SSD_DIR/output -l $SSD_DIR
/logs
#
# save the output and the logs in a tar compressed file
# (assumes wow did not change current working directory)
# otherwise insert: cd $SSD_DIR
tar -cfz /pool/genomics/smart1/great/project/wild-cat-results.tgz output/ logs/
#
# remove everything (in $SSD_DIR), or remove what you know you can (conservative option)
rm -rf *
#
echo $JOB_NAME done `date`
```

# Usage Monitoring Tools

- We have two tools to monitor SSD usage, one on a per-job basis, and one to view usage summary.
- Both need you to load the `gnuplot` and `tools/local` modules

## Per Job Basis

**plot-qssduse.pl**

```
% module load tools/local
% module load gnuplot
% plot-qssduse.pl -x 7420073
```

This example plots the SSD usage of job 7420073 to the screen, assuming you have an X-windows capable connection,

- drop the `-x` to plot to a file,
- and use `NNNN.TTT`, instead of `NNNN` to show usage for a given task (`TTT`) of a job array (`NNNN`).
- Try `plot-qssduse.pl -help` or `man plot-qssduse.pl` for more information.

## Usage Summary

**plot-qssduse-summary.pl**

```
% module load tools/local
% module load gnuplot
% plot-qssduse-summary.pl -x
```

As above:

- drop the `-x` to plot to a file.
- Try `plot-qssduse-summary.pl -help` or `man plot-qssduse-summary.pl` for more information.

---

Last Updated 19 Jul 2019 SGK