Slurm and Singularity Training

for AI cloud II

Jan 2022

CLAAUDIA, Aalborg University

Outline

Background

System design

Getting started

Slurm basics

Working with Singularity images

Tools, tips and tricks

Fair usage

Where to go from here

Background

System design

Getting started

Slurm basics

Working with Singularity images

Tools, tips and tricks

Fair usage

Where to go from here

Background I

- For research purposes at AAU.
- Allow students based on recommendation from staff/supervisor/researcher.
- Free—but we do observe accounting.

Background II

- Update

Background

System design

Getting started

Slurm basics

Working with Singularity images

Tools, tips and tricks

Fair usage

Where to go from here

High level design

- Need new drawing!

Background

System design

Getting started

Slurm basics

Working with Singularity images

Tools, tips and tricks

Fair usage

Where to go from here

Essential skill set and tool set

- Basic Linux, and Shell environment, preferably bash scripting language
- Terminal:
  - Windows: MobaXterm (https://mobaxterm.mobatek.net/)
  - MacOS default terminal or iTerm2 (https://www.iterm2.com/)
  - Linux: Gnome terminal, KDE konsole
- Shell: bash (default), zsh (more feature-rich)

Optional: Byobu (Tmux's User-friendly Wrapper for Ubuntu)

- Benefits:
  - Disconnect from the server while your programs are running in interactive mode.
  - Multiple panel, windows (tabs)
- Byobu demo: activate, Remember Shift + F1, F12 + ?

  create windows and split, switching windows, split
- Alternatives: screen, tmux

Log on the server

- Inside AAU network (on campus or inside VPN):

# One-step log on

```
ssh <aau ID>@ai-fe02.srv.aau.dk
```

Log on the server

- Outside AAU network (outside VPN)

```
# Two-step log on
ssh sshgw.aau.dk -l <aau ID>
Type in your passcode and your Microsoft verification code
ssh ai-fe02.srv.aau.dk -l <aau ID>
# Tunneling
ssh -L 2022:ai-fe02.srv.aau.dk:22\
  -l <aau ID> sshgw.aau.dk
Type in your passcode and your Microsoft verification code
scp -P 2022 ~/Download/testfile <aau ID>@localhost:~/
```

Background

System design

Getting started

Slurm basics

Working with Singularity images

Tools, tips and tricks

Fair usage

Where to go from here

Why?

- Why Slurm?
  - Resource management.
  - Transparency, fairness
  - Widely used. Used before at AAU.

Resource management

- What to manage (resources): walltime, number of GPUs, number of CPUs, memory, ...
- Important management concepts in Slurm terms:
  - Account and organization: cs, es, es.shj
  - Quality of service (QoS): normal, 1gpulong, ...
  - Queuing algorithm: Multi-factor priority plugin

Essential commands

- `sbatch` – Submit job script (batch mode)
- `salloc` – Create job allocation (one option for interactive mode)
- `srun` – Run a command within a batch allocation that was created by sbatch or salloc (or allocates if sbatch or salloc was not used)
- `scancel` – Cancel submitted/running jobs
- `squeue` – View the status of the queue
- `sinfo` – View information on nodes and queues
- `scontrol` – View (or modify) state, e.g. jobs
- `sprio` – View priority computations on queue

Interactive jobs

Two variants (not equivalent)

- `srun --pty --time=20:00 bash -l`

or

- `salloc --time=20:00 --nodelist=nv-ai-03.srv.aau.dk`
- `ssh nv-ai-03.srv.aau.dk`

Slurm Batch job script

```
#!/usr/bin/env bash
#SBATCH --job-name MySlurmJob
#SBATCH --partition batch # equivalent to PBS batch
#SBATCH --time 24:00:00 # Run 24 hours
##SBATCH --gres=gpu:1 # commented out
#SBATCH --qos=normal # examples short, normal, 1gpulong, al
srun echo hello world from sbatch on node $SLURM_NODELIST
```

Submit job:

```
  sbatch jobscript.sh
```

Looking up things and cancel

Basics:

```
sinfo
squeue
scontrol show node
scontrol -d show job <JOBID>
```

Cancelling a job or job step: `scancel`

```
scancel <jobid>
```

Accounting commands

- sacct - report accounting information by individual job and job step
- sreport - report resource usage by cluster, partition, user, account, etc

```
sacct -j 82563 --format=User,JobID,Jobname,partition,\
  state%30,time,start,end,elapsed,qos,ReqMem,AllocGRES
sreport -tminper cluster utilization --tres="gres/gpu" \
  start=9/01/20
```

Slurm query commands

sacctmgr - database management tool

```
sacctmgr show qos \
   format=name,priority,maxtresperuser%20,MaxWall
sacctmgr show assoc format=account,user%30,qos%40
sudo sacctmgr modify user <user> set QOS+=deadline
```

Follow the guidelines on the documentation page and submit an email to support@its.aau.dk if you have a paper deadline.

Slurm: Hints for more advanced uses

Some additional readings:

- Trackable Resource
- Accounting
- Resource Limit
- Dependencies
- Job array
- Information for developers
- run `man <commandname>` for builtin documentation. For example: `man scontrol`

Background

System design

Getting started

Slurm basics

Working with Singularity images

Tools, tips and tricks

Fair usage

Where to go from here

Singularity overview

1. You get your own environment.
   - Flexibility in software
   - Flexibility in version
   - User requests/changes does not affect others
   - Draw on the Docker images NVIDIA supplies in their NGC (can convert Docker to Singularity image)
2. Security: overcome Docker's drawbacks
   - root access / UID problem
   - resource exposure
3. Compatibility with Slurm
4. HPC-oriented
5. Users familar with Docker might experience slow build process.

Refs Docker vs. Singularity discussion: ref and ref2

Check built-in documentation

```
srun singularity -h
```

see `singularity help <command>`

Singularity build from Docker and exec command

Example: Pull a Docker image and convert to singularity image

```
srun singularity pull docker://godlovedc/lolcow
```

and then run

```
srun singularity run lolcow_latest.sif
```

Singularity build from NGC Docker image

```
srun singularity pull
docker://nvcr.io/nvidia/tensorflow:20.10-tf2-py3
```

- Takes some time. Placed a copy in:

/user/share/singularity/images/tensorflow_20.10-tf2-py3.sif

Singularity build from NGC Docker image

- Common use case for interactive work:

```
srun --pty --gres=gpu:1 singularity shell --nv
tensorflow_20.10-tf2-py3.sif

nvidia-smi
ipython
import tensorflow
exit
exit
```

- With the last exit you released the resources. Keep connection alive using tmux, screen, or byobo to avoid releasing.

Combining all the steps from today

Example:

```
srun --gres=gpu:1 singularity exec --nv \
-B .:/code -B mnist-data/:/data -B output_data:/output_data
tensorflow_20.10-tf2-py3.sif python /code/example.py 10
```

Can be inserted in a batch script

Build a customized Singularity images

Singularity definition file

Example: Singularity

```
BootStrap: docker
From: nvcr.io/nvidia/tensorflow:20.10-tf2-py3

%post
pip install keras
```

You can then build with

```
srun singularity build --fakeroot \
  tensorflow_keras.sif Singularity
```

Running your customized Singularity images

You can then run a specific command with

```
srun --gres=gpu:1 singularity exec --nv \
  -B .:/code -B output_data:/output_data \
  tensorflow_keras.sif python /code/example.py 10
```

or enter an interactive session with

```
srun --pty --gres=gpu:1 singularity shell --nv \
  -B .:/code -B output_data:/output_data \
  tensorflow_keras.sif
```

Background

System design

Getting started

Slurm basics

Working with Singularity images

Tools, tips and tricks

Fair usage

Where to go from here

Tools, tips and tricks I

On the node:

- View resource utilization on compute node (shh in):
  - `$ top -u <user>`
  - `$ smem -u -k`
  - `$ nvidia-smi -l 1 -i <IDX>` # see scontrol -d show job
- View GPU resource utilization from front-end node
  - `$ getUtilByJobId.sh <JobId>`
- Data in e.g. /user/student.aau.dk/ are on a distributed file system
  - Consider using /raid (SSD NVMe) on the compute node (see doc)
- If you have allocated a GPU and your job information contains `mem=10000M` and it is just pending (state=PD, possible reason=resources) but there should be resources.
  - Issue: cancel and add e.g. –mem=64G to you allocation

Tools, tips and tricks II

Things to consider in your framework:

- On the system: 6 CPUs and ~90G per GPU on average.
  - Consider scaling workers (in framework) and CPUs.
  - In general we run out of GPUs first, then memory. Consider adding more CPUs to push jobs through.
- V100 tensor cores are half-precision float. For speed: use half-precision or mixed precision.
- The DGX-2 comes with NVLink+NVSwitch: increase bandwidth between GPUs and allow for efficient multi-GPU programming.

Background

System design

Getting started

Slurm basics

Working with Singularity images

Tools, tips and tricks

Fair usage

Where to go from here

Fair usage

We kindly ask that all users consider the following guidelines:

- Please be mindful of your allocations and refrain from allocating many resources without knowing/testing/verifying that you can indeed make good use of the allocated resources.
- Please be mindful and de-allocate the resources if you do no use them. Then other users can make good use of these.

We see challenges towards the end of semesters (cyclic):

- More HW (NVIDIA T4) is on the way in.
- It is for research ... administration intend to interfere as little as possible ... but we do try to help and do something.
- Resource discussion in the steering committee — contact your faculty representative.

Background

System design

Getting started

Slurm basics

Working with Singularity images

Tools, tips and tricks

Fair usage

Where to go from here

Where to go from here

- The user documentation
  - More workflows
  - Copying data to the local drive for higher I/O performance
  - Inspecting your utilization
  - Matlab, pytorch, ...
  - Fair usage/upcoming deadline
  - Links and references to additional material
  - Support (fastest response): support@its.aau.dk
  - Advisory (slower response – longer time span): claaudia@aau.dk
- Use the resource and give feedback. Share with us your success stories (including benchmarks, solved challenges, new possibilities, etc.)
- Share with other users on the Yammer channel.