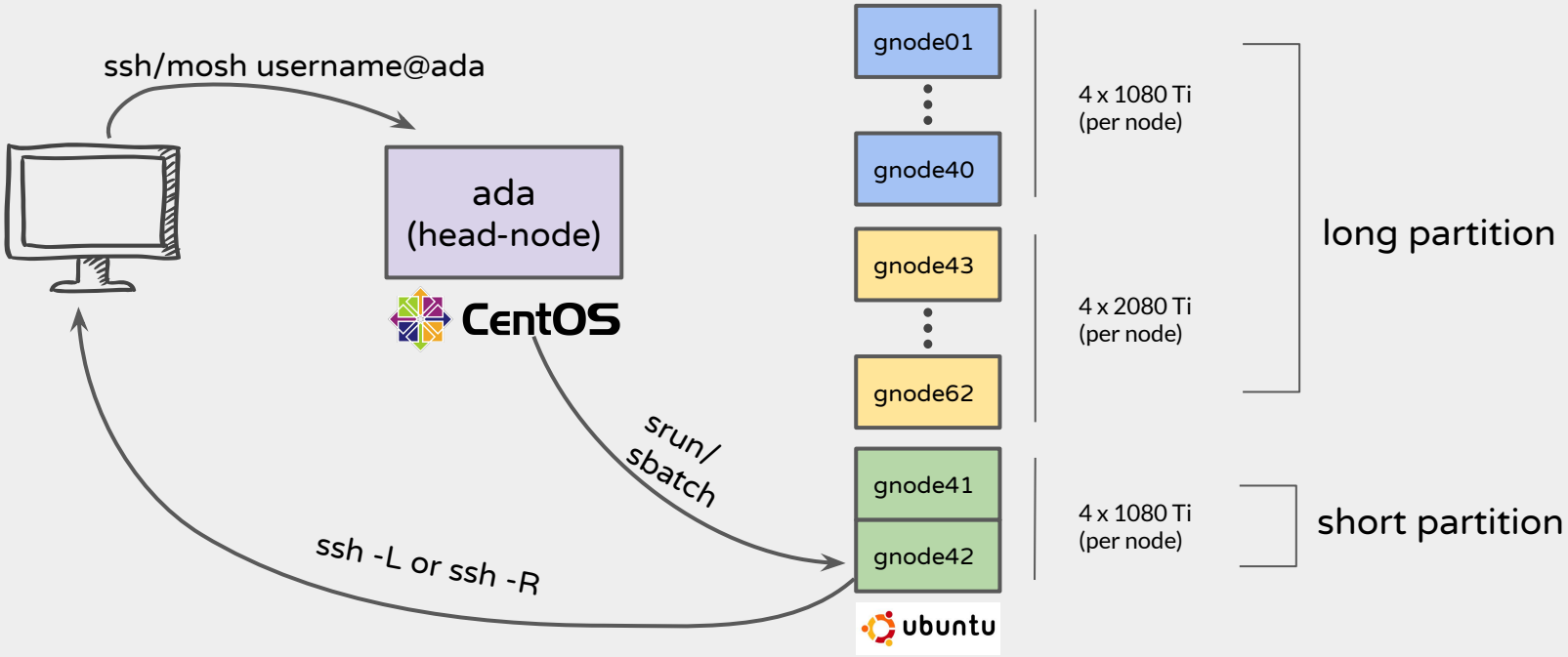
$ ada-tutorial

Slides by Rohan Chacko & Jerin Philip

$ init_

- ada is your compute cluster. All your GPU-intensive jobs will run on this.
- tutorial is linux-oriented, entire ada ecosystem is as well.
- cluster is managed by slurm - a job scheduling and cluster management system
- each student has a separate slurm account. Consider it like a bank account with gpu mins as currency. Use it wisely.

$ model_


ssh/mosh username@ada
ada
(head-node)
CentOS
srun/
sbatch
ssh -L or ssh -R
gnode01
gnode40
gnode43
gnode62
gnode41
gnode42
ubuntu
4 x 1080 Ti
(per node)
4 x 2080 Ti
(per node)
4 x 1080 Ti
(per node)
long partition
short partition

$ ~~ssh~~ mosh

> Install

sudo apt-get install mosh (Debian)

brew install mosh (MacOS)

> Login

youngling@ada.iiit.ac.in

or youngling@10.4.24.24

passwd: **BabyYodaBest**

$ partitions_

```
[youngling@ada ~]$ sinfo

PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
short        up    6:00:00      2   idle gnode[41-42]
long*        up   infinite     20    mix gnode[01,11,23..]
long*        up   infinite     17  alloc gnode[03-04,15,22..]
long*        up   infinite     21  drain gnode[16,38..]
long*        up   infinite      2  maint gnode[05-06]
```

$ nodestates_

STATE	MEANING
IDLE	free to use
ALLOCATED	fully allocated
MIXED	have some CPUs free
MAINT	under maintenance
DRAIN	currently unavailable for use

these are the common ones. more are mentioned in the slurm pages

$ account-limits

```
# Association Limits
sacctmgr show assoc                          \
 User="" Account=youngling                   \
 format=Account,GrpTRES,GrpTRESMins -p \
    | column -t -s "|";

# Usage
scontrol show assoc_mgr                      \
  Account=youngling flags=assoc              \
    | grep --color=auto -m 1 'GrpTRESMins'
```

$ storage

A youngling gets:

- 25GB on /home/youngling
- **50** (default) | 100 | 150 | 200 GB on /share3
- more storage can be allocated for special cases

use /share3 for long-term file storage like checkpoints, logs, etc.

Common datasets are curated and mounted at /share1/dataset

$ storage

In gnodeXY, you get write access on

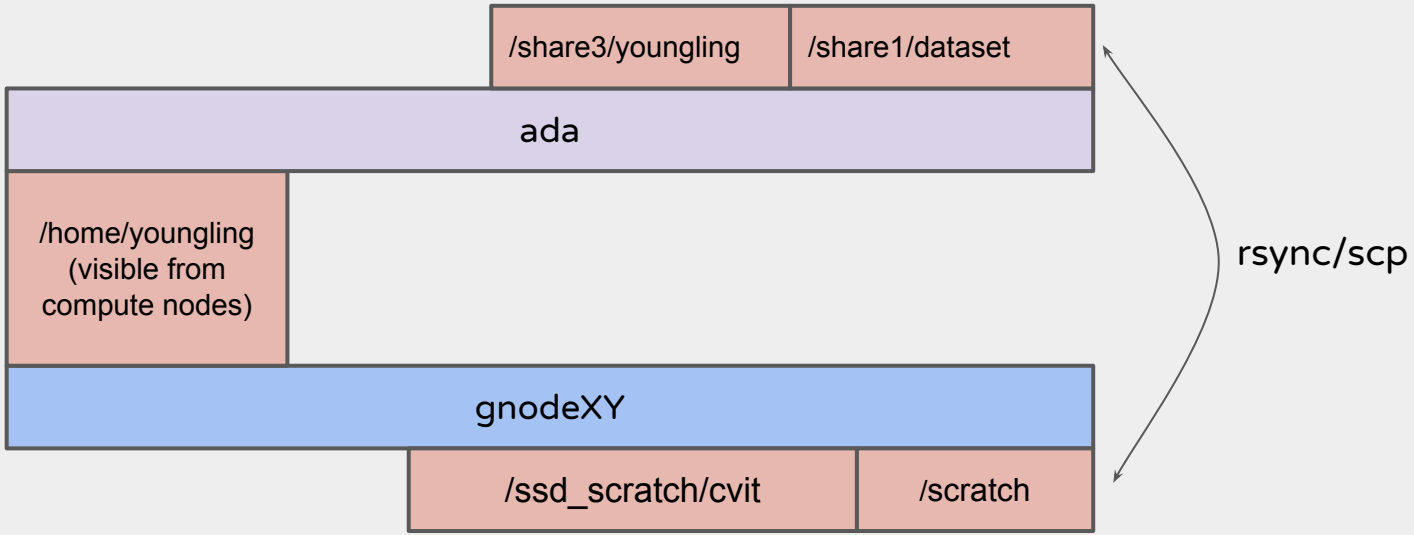
- /ssd_scratch/cvit - lasts 10 days, SSD, hence faster
- /scratch - lasts 10 days

for intermediate data storage when running models

$ storage-summary

	SPACE	QUOTA	VISIBILITY	PURPOSE	DELETION POLICY	BACKUP
headnode	/home	25 GB per user	head-node & gnode	small files, anaconda	-	Yes (daily)
	/share1	6TB (whole cvit)	only head-node	public datasets	-	No
	/share3	50 GB per user	only head-node	long-term storage	-	No
gnode	/scratch	2 TB (total)	only gnodeXY	temp hdd storage	10 days	No
	/scratch_ssd	960GB (total)	only gnodeXY	temp ssd storage for fast I/O	10 days	No

$ typical data workflow



nfs: Network File System

$ warning

- Do not run heavy jobs on ada (head-node). Use it only to ssh in and gain access.
- Compile / Install packages only after gaining allocation on gnode. The OS mismatch will lead to your packages not working on gnodes (Ubuntu) if you install them on head-node (CentOS).
- /home/$USER has a limit on number of files - 300k total
- Create folders nested under your name to identify what is yours. There are many concurrent users.
- 3 failed login attempts and you're locked out

[hands-on]

$ submit-job

- On ada, you submit “jobs” to gain access to a gnode
- Either use an interactive session (srun) or a headless session (sbatch)
- In either modes, mention the resource requirements followed by the command(s) to run on gaining access
- Avoid running interactive sessions on long partitions. Your idle time coding could be somebody else’s gpu time. Gain interactive shells on short partitions (6:00:00 limit)
- Your session will be terminated if you try to gain access without a GPU

$ module-load

- Environment modules allows users to set shell env variables needed to run programs

```
(base) [youngling@ada ~]$ module avail

------------------------------------------------------------------------ /opt/Modules/versions ------------------------------------------------------------------------
3.2.10

------------------------------------------------------------------- /opt/Modules/3.2.10/modulefiles -------------------------------------------------------------------
amber/18                          cudnn/7.6.4-cuda-9.0              glog/0.4.0                        module-info                       openmpi/4.0.0
capnproto/0.6.1                   cudnn/7.6.5-cuda-10.2             gromacs/2016.3                    modules                           openmpi/4.0.1-cuda10
capnproto/0.7.0                   cudnn/7.6-cuda-10.0               gromacs/2016.3-plumed             mpich/3.3.1                       pcl/1.8.1
ceres-solver/1.14.0-165-gd7f428e  cudnn/7-cuda-10.0                 gromacs/2019                      mrtrix/3.0                        plumed/2.5.2
cmake/3.15.2                      cudnn/7-cuda-9.0                  gromacs/2019.3-plumed             namd/2.12                         python/3.6.8
colmap/3.6-dev.2-15-g6b6e825      dot                               lammps/7Aug19                     namd/2.13                         python/3.7.4
cuda/10.0                         eigen/3.3.7                       lammps/7Aug19-v2                  null                              R/3.6.2
cuda/10.1                         ffmpeg/4.0.1                      leptonica/1.78.0                  openblas/0.3.6                    singularity/2.5.2
cuda/10.2                         ffmpeg/4.2.1                      matlab/R2019b                     opencv/4.1.2                      tbb/2018u1-debug
cuda/9.0                          freesurfer/6.0.0                  matlab/R2020a                     opencv/4.2.0                      tbb/2018u1-release
cuda/9.1                          gflags/2.2.1                      mkl/2019.3.199                    openmesh/8.0                      tesseract/4.1.0
cudnn/7.1-cuda-9.1                gflags/2.2.2                      mkl/2019.4.243                    openmpi/2.1.1                     use.own
cudnn/7.3-cuda-10.0               glog/0.3.5                        module-git                        openmpi/3.1.0                     VTK/8.2.0
```

`module avail` - list all modules available

`module load <module-name>` - load module

> running-jobs_

- srun
- sbatch

$ srun ...

```
> srun
    --pty                          # interactive
    --account youngling            # [<user>|research|sub]
    --time 4-00:00:00              # [days]-[hh]:[mm]:[ss]
    /bin/bash -l
```

$ simple-jobs_

But defaults aren't usually nice to you! :(

```
> srun
    --pty                       # interactive
    --account youngling
    --partition long            # [short|long]
    --time 4-00:00:00           # [days]-[hh]:[mm]:[ss]
    --nodes 1                   # cpu cores
    --gres=gpu:0                # gpus
    /bin/bash -l
```

$ limits/upgrade

- cvit-queue is meant to be high availability and users expected to optimally use resources.
- Pay for your GPU usage by GPUMins. Your GPUMins is replenished start of every month. Use it wisely.
- GPUMins consumed = no. of GPUs used x total usage minutes (begins from allocation of gnode)
- initially each user is restricted to 1 GPU for 600 GPUmins/week. your usage pattern will be considered while upgrading.
  - prefer batch jobs to interactives

$ some-tips_

prefer long-option arguments to the short-hands.
the latter can be confusing.

--nodes -N
--ntasks -n

use the following proportions:

1 gpu : **10** cores : **2-3GB** mem-per-cpu
(4 gpu: 40 cores : 120GB mem-per cpu ~ 1 node)

$ some-tips_

Keep your environment clean. Use conda/virtualenv

Make a habit of using environments for your projects rather than installing packages at default /home/$USER/bin

Avoids conflicting packages and is easier to clone on another system

$ some-tips_

.tar your large files

Always tar your large files (datasets, etc) as this significantly increases copy speed by eliminating file-creation overheads.

Untar them at the nodes.

NOT tar.gz (you don’t want to compress uncompress).

Time to tinker

Create a small file in ada:/share3/youngling

Gain a bash shell through srun, try copying in and out of.

- /scratch
- /ssd_scatch/cvit

$ inspect

```
# inspect variables set by SLURM
env | grep "SLURM"

# check properties of a specific job
scontrol show jobs <jobid>

# search
scontrol show jobs -o | grep "<regex>"

squeue -u $USER # only your jobs
```

$ why am i not getting an alloc?

There could be many reasons shown:

- **Priority**
  There are jobs waiting to be scheduled ahead of you in the queue. Ada's scheduling is FCFS.

- **Resources**
  There are not enough resources to allocate to you.

$ why am i not getting an alloc?

- **AssocGrpGres**
  Your group (cvit, ccnsb, research etc) is already using the max possible GPUs allocated to it. At max 60 GPUs can be used by all users belonging to the cvit user-group. Use `sinfo.x` command to check this.

- **AssocGrpCpuLimit | AssocGrpMemLimit**:
  Same as above, except for CPU or Memory.

- **AssocGrpGresMins**
  You have exhausted the *GPUMins* for the month. If you want it increased, contact cvit-admins.

$ when will I get an allocation?

scontrol show job <JobID> | grep -o StartTime=[^\ ]*

$ sbatch

can run headless.

scripts:
- setup to save stdout/stderr -> experiments are logged.
- commands are saved and documented.

mail notifications when complete.

sbatch is how you can and should run distributed jobs.

$ sbatch-script-skeleton

```
#!/bin/bash

# SBATCH ...

module load ...      #
rsync ... or scp .. # Copy-In Data
function _export {<save-checkpoint-work>}
trap "_export" SIGXXX

# Code running

wait
_export
```

$ sbatch

```
#!/bin/bash
#SBATCH --account youngling
#SBATCH --nodes 1
#SBATCH --cores 8
#SBATCH --gres=gpu:1
#SBATCH --mem-per-cpu=2G
#SBATCH --time=1-00:00:00
#SBATCH --mail-type=END
#SBATCH --mail-user=<mail-id>

# Below command echoes commands as they're
# executed.
set -x;

echo “Running on gnode” >> ~/<dir>/log.txt
```

$ fault-tolerance > saving | resuming

```
#SBATCH --signal=B:TERM@900

CHECKPOINTS=...
REMOTE_DIR=...

function _export {
    ssh $USER@ada "mkdir -p $REMOTE_DIR/"          # create-remote-directory
    rsync -rz                                     \ # rsync
        $CHECKPOINTS/checkpoint_{best,last}.pt    \ # src
        ada:$REMOTE_DIR/                            # dest
}


## We trap the SIGHUP signal and register a handler
## which saves computed checkpoint
## in this case

trap "_export" TERM
```

scancel *<JOB-ID>* - cancel currently running jobs

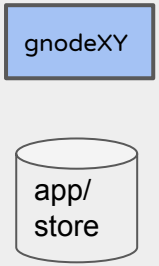
$ more interactive-stuff

remote-port-forwarding

Many a times, during prototyping it's convenient to bind ports of a gnode to your local-machine.

e.g:

- running a jupyter-notebook on ada
- running tensorboard on ada



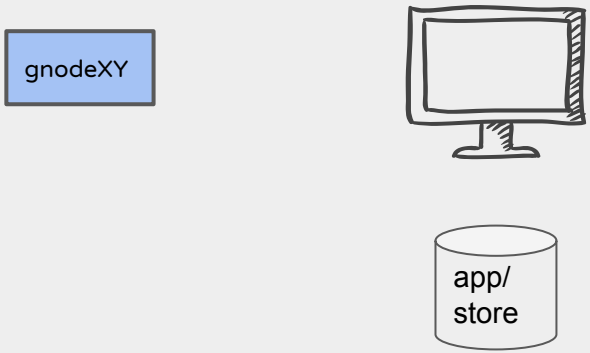
$ remote-port-forwarding > jupyter

```
jupyter-notebook      \
    --ip localhost    \
    --port 8888       \
    & # background

ssh
   -N # Do not execute a remote command
   -f # Background, so we can proceed
   -R # Remote Port Forwarding
    8888 # your-machine[bind_address:port]
    :localhost:8888  # gnode[bind_address:port]
    username@10.2.16.xx  # where to ssh and set-this-up.
```

tensorboard? sftp?

local-port forwarding



Often, it makes sense to do the other way around, say for logging setups like visdom

This way, even if your job exited, your logs are saved on your local machine for you to inspect and won't have to fight other people for the accessing same node.

$ local-port-forwarding

```
# default - localhost:8097
[youngling@gnodexx]$ tensorboard --logdir=expt_1.0 &

[youngling@gnodexx]$
    ssh
       -N # Do not execute a remote command
       -f # Background, so we can proceed
       -L # Local Port Forwarding
      8097:localhost:8097
      username@ip # ip given by vpn(tun0). Starts with 10.2...
```

$ tips & tricks

- Mosh : handles connection timeouts gracefully. Instead of `ssh username@ip` -> `mosh username@ip`
- tmux : run multiple shell instances without losing state. Will help when training models over hours/days
- You can use sh/bash/zsh on ada. Call the executable accordingly.
- If you want to run matlab or other interactive apps on ada, ssh with -X flag and load corresponding modules

$ tmux - terminal multiplexer

- Basic commands
  - tmux new -s <session-name> `# Creates new tmux session`
  - tmux a -t <session-name> `# Attach to a running session`
  - prefix(ctrl+b) + d `# Detach from session`
  - tmux ls `# List all sessions`
  - tmux kill-ses -t <session-name> `# Kill session`

For more commands - `man tmux`
You can also find tmux cheat sheets online - Link

Homework(optional): tmux on ada (headnode) is old. Provide a build script so we can put one in cvit-contrib/modules for everyone.

$ tips & tricks

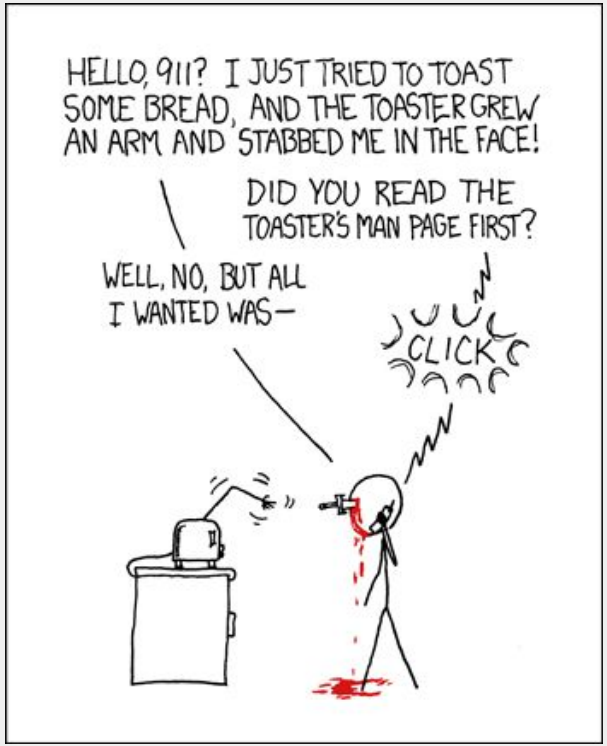
CUDA memory error? Not the end of the world.

Try :

- Make sure you are not performing unnecessary gradient computations
- Reduce batch size
- Acquire more GPUs and parallelize across them
  > model = nn.DataParallel(model)
  > device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
  > model.to(device)

$ man when in doubt

- all commands have extensive man pages. Get familiar with it.
- SLURM docs is your friend - Slurm Workload Manager - Man Pages
- you will run into errors, you have to send a paste of the log. ix.io is recommended. pipe error/output from ada shell and mail admins.



$ other-resources_

- IIIT Ada User Guide - http://hpc.iiit.ac.in/wiki/index.php/Ada_User_Guide
- Slurm docs - https://slurm.schedmd.com/overview.html
- Video tutorial by cvit-admins - Video link
- Sample sbatch script - Github Gist

For any issues, mail cvit-admins - cvit-sudo@googlegroups.com

$ Homework

- Figure out how to use tensorboard for your models (pytorch uses tensorboardx...already installed on youngling)
- Parallelize a CNN model from previous class using the data parallelism method you have studied. See if losses change with increase in batch-size (1, 2, 4, 8..)

Use youngling to experiment until you receive separate ada accounts.

Thank you!