Speed: The GCS ENCS Cluster

Serguei A. Mokhov Gillian A. Roper Carlos Alarcón Meza Farah Salhany
Network, Security and HPC Group*
Gina Cody School of Engineering and Computer Science
Concordia University
Montreal, Quebec, Canada
rt-ex-hpc~AT~encs.concordia.ca

Version 7.3

Abstract

This document serves as a quick start guide to using the Gina Cody School of Engineering and Computer Science (GCS ENCS) compute server farm, known as "Speed." Managed by the HPC/NAG group of the Academic Information Technology Services (AITS) at GCS, Concordia University, Montreal, Canada.

Contents

*The group acknowledges the initial manual version VI produced by Dr. Scott Bunnell while with us as well as Dr. Tariq Daradkeh for his instructional support of the users and contribution of examples.

1 Introduction

This document contains basic information required to use "Speed", along with tips, tricks, examples, and references to projects and papers that have used Speed. User contributions of sample jobs and/or references are welcome.

Note: On October 20, 2023, we completed the migration to SLURM from Grid Engine (UGE/AGE) as our job scheduler. This manual has been updated to use SLURM's syntax and commands. If you are a long-time GE user, refer to Appendix A.2 for key highlights needed to translate your GE jobs to SLURM as well as environment changes. These changes are also elaborated throughout this document and our examples.

1.1 Citing Us

If you wish to cite this work in your acknowledgements, you can use our general DOI found on our GitHub page `https://dx.doi.org/10.5281/zenodo.5683642` or a specific version of the manual and scripts from that link individually. You can also use the "cite this repository" feature of GitHub.

1.2 Resources

- Public GitHub page where the manual and sample job scripts are maintained at `https://github.com/NAG-DevOps/speed-hpc`
  - Pull requests (PRs) are subject to review and are welcome: `https://github.com/NAG-DevOps/speed-hpc/pulls`
- Speed Manual:
  - PDF version of the manual: `https://github.com/NAG-DevOps/speed-hpc/blob/master/doc/speed-manual.pdf`
  - HTML version of the manual: `https://nag-devops.github.io/speed-hpc/`
- Concordia official page for "Speed" cluster, which includes access request instructions. `https://www.concordia.ca/ginacody/aits/speed.html`
- All Speed users are subscribed to the `hpc-ml` mailing list.

1.3 Team

Speed is supported by:

- Serguei Mokhov, PhD, Manager, Networks, Security and HPC, AITS
- Gillian Roper, Senior Systems Administrator, HPC, AITS
- Carlos Alarcón Meza, Systems Administrator, HPC and Networking, AITS
- Farah Salhany, IT Instructional Specialist, AITS

We receive support from the rest of AITS teams, such as NAG, SAG, FIS, and DOG.

`https://www.concordia.ca/ginacody/aits.html`

1.4 What Speed Consists of

- Twenty four (24) 32-core compute nodes, each with 512 GB of memory and approximately 1 TB of local volatile-scratch disk space (pictured in Figure 1).
- Twelve (12) NVIDIA Tesla P6 GPUs, with 16 GB of GPU memory (compatible with the CUDA, OpenGL, OpenCL, and Vulkan APIs).
- 4 VIDPRO nodes (ECE. Dr. Amer), with 6 P6 cards, and 6 V100 cards (32GB), and 256GB of RAM.
- 7 new SPEED2 servers with 256 CPU cores each 4x A100 80 GB GPUs, partitioned into 4x 20GB MIGs each; larger local storage for TMPDIR (see Figure 2).
- One AMD FirePro S7150 GPU, with 8 GB of memory (compatible with the Direct X, OpenGL, OpenCL, and Vulkan APIs).
- Salus compute node (CSSE CLAC, Drs. Bergler and Kosseim), 56 cores and 728GB of RAM, see Figure 2.
- Magic subcluster partition (ECE, Dr. Khendek, 11 nodes, see Figure 2).
- Nebular subcluster partition (CIISE, Drs. Yan, Assi, Ghafouri, et al., Nebulae GPU node with 2x RTX 6000 Ada 48GB cards, Stellar compute node, and Matrix 177TB storage/compute node, see Figure 2).

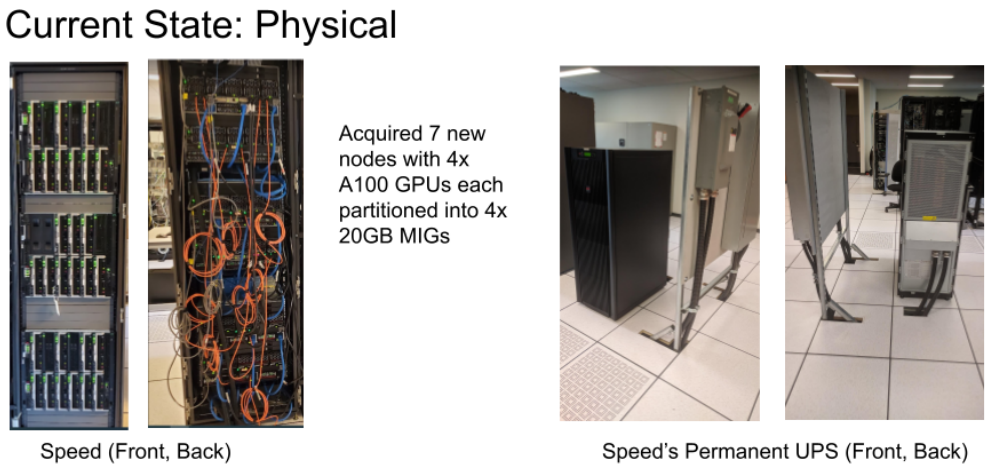

Figure 1: Speed

1.5 What Speed Is Ideal For

- Design, develop, test, and run parallel, batch, and other algorithms and scripts with partial data sets. "Speed" has been optimized for compute jobs that are multi-core aware,

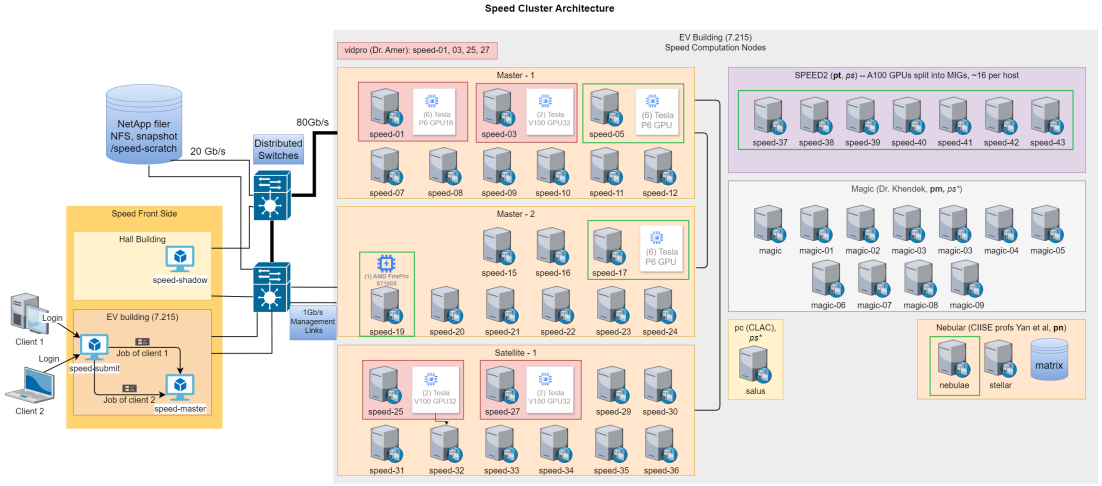

Figure 2: Speed Cluster Hardware Architecture

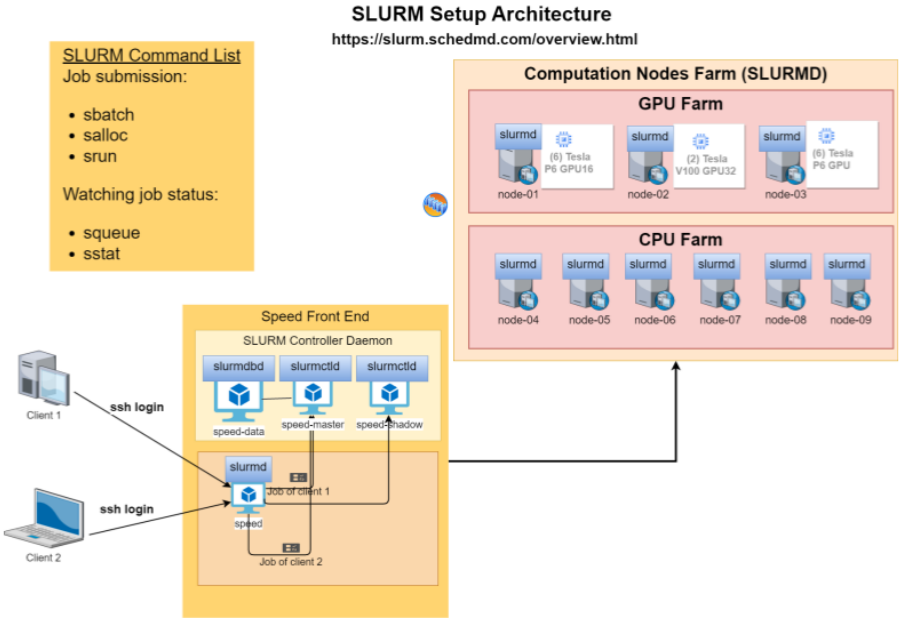

Figure 3: Speed SLURM Architecture

require a large memory space, or are iteration intensive.

- Prepare jobs for large clusters such as:
  - Digital Research Alliance of Canada (Calcul Quebec and Compute Canada)
  - Cloud platforms
- Jobs that are too demanding for a desktop.
- Single-core batch jobs; multithreaded jobs typically up to 32 cores (i.e., a single machine).
- Multi-node multi-core jobs (MPI).
- Anything that can fit into a 500-GB memory space and a **speed scratch** space of approximately 10 TB.
- CPU-based jobs.
- CUDA GPU jobs.
- Non-CUDA GPU jobs using OpenCL.

1.6 What Speed Is Not

- Speed is not a web host and does not host websites.
- Speed is not meant for Continuous Integration (CI) automation deployments for Ansible or similar tools.
- Does not run Kubernetes or other container orchestration software.
- Does not run Docker. (**Note:** Speed does run Singularity and many Docker containers can be converted to Singularity containers with a single command. See Section 2.16.)
- Speed is not for jobs executed outside of the scheduler. (Jobs running outside of the scheduler will be killed and all data lost.)

1.7 Available Software

There are a wide range of open-source and commercial software available and installed on "Speed." This includes Abaqus [1], AllenNLP, Anaconda, ANSYS, Bazel, COMSOL, CPLEX, CUDA, Eclipse, Fluent [2], Gurobi, MATLAB [15, 29], OMNeT++, OpenCV, OpenFOAM, OpenMPI, OpenPMIx, ParaView, PyTorch, QEMU, R, Rust, and Singularity among others. Programming environments include various versions of Python, C++/Java compilers, TensorFlow, OpenGL, OpenISS, and MARF [30].

In particular, there are over 2200 programs available in `/encs/bin` and `/encs/pkg` under Scientific Linux 7 (EL7). We are building an equivalent array of programs for the EL9 SPEED2 nodes. To see the packages available, run `ls -al /encs/pkg/` on `speed.encs`. See a complete list in Appendix D.

Note: We do our best to accommodate custom software requests. Python environments can use user-custom installs from within scratch directory.

1.8 Requesting Access

After reviewing the "What Speed is" (Section 1.5) and "What Speed is Not" (Section 1.6), request access to the "Speed" cluster by emailing: `rt-ex-hpc AT encs.concordia.ca`.

- GCS ENCS faculty and staff may request access directly.
- GCS students must include the following in their request message:
  - GCS ENCS username
  - Name and email (CC) of the approver – either a supervisor, course instructor, or a department representative (e.g., in the case of undergraduate or M.Eng. students it can be the Chair, associate chair, a technical officer, or a department administrator) for approval.
  - Written request from the approver for the GCS ENCS username to be granted access to "Speed."
- Non-GCS students taking a GCS course will have their GCS ENCS account created automatically, but still need the course instructor's approval to use the service.
- Non-GCS faculty and students need to get a "sponsor" within GCS, so that a guest GCS ENCS account is created first. A sponsor can be any GCS Faculty member you collaborate with. Failing that, request the approval from our Dean's Office; via our Associate Deans Drs. Eddie Hoi Ng or Emad Shihab.
- External entities collaborating with GCS Concordia researchers should also go through the Dean's Office for approvals.

For detailed instructions, refer to the Concordia Computing (HPC) Facility: Speed webpage.

2 Job Management

We use SLURM as the workload manager. It supports primarily two types of jobs: batch and interactive. Batch jobs are used to run unattended tasks, whereas, interactive jobs are are ideal for setting up virtual environments, compilation, and debugging.

Note: In the following instructions, anything bracketed like, `<>`, indicates a label/value to be replaced (the entire bracketed term needs replacement).

Job instructions in a script start with `#SBATCH` prefix, for example:

```
#SBATCH --mem=100M -t 600 -J <job-name> -A <slurm account>
#SBATCH -p pg --gpus=1 --mail-type=ALL
```

For complex compute steps within a script, use `srun`. We recommend using `salloc` for interactive jobs as it supports multiple steps. However, `srun` can also be used to start interactive jobs (see Section 2.8). Common and required job parameters include:

Common and required job parameters include:

- Memory (`--mem=<mem>[M|G|T]`),
- Partition/Queue (`-p <partition>`),
- Job name (`--job-name=<name>` or `-J <name>`),

- Wall Clock Limit (`-t <min> or -t <days-hh:mm:ss>`),
- Event Notification (`--mail-type=<events>`),
- Email Address (`--mail-user=<address>`),
- Slurm Account (`--account=<account> or -A <account>`),
- Tasks Per Node (`--tasks-per-node=<count>`),
- CPUs Per Task (`--cpus-per-task=<count>`),
- CPU Count ntasks (`-n <count>`).

2.1 Getting Started

Before getting started, please review the "What Speed is" (Section 1.5) and "What Speed is Not" (Section 1.6). Once your GCS ENCS account has been granted access to "Speed", use your GCS ENCS account credentials to create an SSH connection to `speed` (an alias for `speed-submit.encs.concordia.ca`).

All users are expected to have a basic understanding of Linux and its commonly used commands (see Appendix B for resources).

2.1.1 SSH Connections

Requirements to create SSH connection to "Speed":

1. **Active GCS ENCS user account:** Ensure you have an active GCS ENCS user account with permission to connect to Speed (see Section 1.8).
2. **VPN Connection** (for off-campus access): If you are off-campus, you wil need to establish an active connection to Concordia's VPN, which requires a Concordia netname.
3. **Terminal Emulator for Windows:** Windows systems use a terminal emulator such as PuTTY, Cygwin, or MobaXterm.
4. **Terminal for macOS:** macOS systems have a built-in Terminal app or `xterm` that comes with XQuartz.

To create an SSH connection to Speed, open a terminal window and type the following command, replacing `<ENCSusername>` with your ENCS account's username:

```
ssh <ENCSusername>@speed.encs.concordia.ca
```

For detailed instructions on securely connecting to a GCS server, refer to the AITS FAQ: How do I securely connect to a GCS server?

2.1.2 Environment Set Up

After creating an SSH connection to Speed, you will need to make sure the `srun`, `sbatch`, and `salloc` commands are available to you. To check this, type each command at the prompt and press Enter. If "command not found" is returned, you need to make sure your `$PATH` includes `/local/bin`. You can check your path by typing:

```
echo $PATH
```

The next step is to set up your cluster-specific storage "speed-scratch", to do so, execute the following command from within your home directory.

```
mkdir -p /speed-scratch/$USER && cd /speed-scratch/$USER
```

Next, copy a job template to your cluster-specific storage

- From Windows drive G: to Speed:
  `cp /winhome/<1st letter of $USER>/$USER/<script>.sh /speed-scratch/$USER/`
- From Linux drive U: to Speed:
  `cp ~/<script>.sh /speed-scratch/$USER/`

Tip: the default shell for GCS ENCS users is `tcsh`. If you would like to use `bash`, please contact `rt-ex-hpc AT encs.concordia.ca`.

Note: If you encounter a "command not found" error after logging in to Speed, your user account may have defunct Grid Engine environment commands. See Appendix A.2 for instructions on how to resolve this issue.

2.2 Job Submission Basics

Preparing your job for submission is fairly straightforward. Start by basing your job script on one of the examples available in the `src/` directory of our GitHub repository. You can clone the repository to get the examples to start with via the command line:

```
git clone --depth=1 https://github.com/NAG-DevOps/speed-hpc.git
cd speed-hpc/src
```

The job script is a shell script that contains directives, module loads, and user scripting. To quickly run some sample jobs, use the following commands:

```
sbatch -p ps -t 10 env.sh
sbatch -p ps -t 10 bash.sh
sbatch -p ps -t 10 manual.sh
sbatch -p pg -t 10 lambdal-singularity.sh
```

2.2.1 Directives

Directives are comments included at the beginning of a job script that set the shell and the options for the job scheduler. The shebang directive is always the first line of a script. In your job script, this directive sets which shell your script's commands will run in. On "Speed", we recommend that your script use a shell from the `/encs/bin` directory.

To use the `tcsh` shell, start your script with `#!/encs/bin/tcsh`. For `bash`, start with `#!/encs/bin/bash`.

Directives that start with `#SBATCH` set the options for the cluster's SLURM job scheduler. The following provides an example of some essential directives:

```
#SBATCH --job-name=<jobname>        ## or -J. Give the job a name
#SBATCH --mail-type=<type>          ## set type of email notifications
#SBATCH --chdir=<directory>         ## or -D, set working directory for the job
#SBATCH --nodes=1                   ## or -N, node count required for the job
```

```
#SBATCH --ntasks=1                  ## or -n, number of tasks to be launched
#SBATCH --cpus-per-task=<corecount> ## or -c, core count requested, e.g. 8 cores
#SBATCH --mem=<memory>              ## assign memory for this job,
                                    ## e.g., 32G memory per node
```

Replace the following to adjust the job script for your project(s)

- `<jobname>` with a job name for the job. This name will be displayed in the job queue.
- `<directory>` with the fullpath to your job's working directory, e.g., where your code, source files and where the standard output files will be written to. By default, `--chdir` sets the current directory as the job's working directory.
- `<type>` with the type of e-mail notifications you wish to receive. Valid options are: NONE, BEGIN, END, FAIL, REQUEUE, ALL.
- `<corecount>` with the degree of multithreaded parallelism (i.e., cores) allocated to your job. Up to 32 by default.
- `<memory>` with the amount of memory, in GB, that you want to be allocated per node. Up to 500 depending on the node.
  Note: All jobs MUST set a value for the `--mem` option.

Example with short option equivalents:

```
#SBATCH -J myjob              ## Job's name set to 'myjob'
#SBATCH --mail-type=ALL       ## Receive all email type notifications
#SBATCH -D ./                 ## Use current directory as working directory
#SBATCH -N 1                  ## Node count required for the job
#SBATCH -n 1                  ## Number of tasks to be launched
#SBATCH -c 8                  ## Request 8 cores
#SBATCH --mem=32G             ## Allocate 32G memory per node
```

Tip: If you are unsure about memory footprints, err on assigning a generous memory space to your job, so that it does not get prematurely terminated. You can refine `--mem` values for future jobs by monitoring the size of a job's active memory space on `speed-submit` with:

```
sacct -j <jobID>
sstat -j <jobID>
```

This can be customized to show specific columns:

```
sacct -o jobid,maxvmsize,ntasks%7,tresusageouttot%25 -j <jobID>
sstat -o jobid,maxvmsize,ntasks%7,tresusageouttot%25 -j <jobID>
```

Memory-footprint efficiency values (`seff`) are also provided for completed jobs in the final email notification as "maxvmsize". *Jobs that request a low-memory footprint are more likely to load on a busy cluster.*

Other essential options are `--time`, or `-t`, and `--account`, or `-A`.

- `--time=<time>` – is the estimate of wall clock time required for your job to run. As previously mentioned, the maximum is 7 days for batch and 24 hours for interactive jobs. Jobs with a smaller `time` value will have a higher priority and may result in your job being scheduled sooner.

- `--account=<name>` – specifies which Account, aka project or association, that the Speed resources your job uses should be attributed to. When moving from GE to SLURM users most users were assigned to Speed's two default accounts `speed1` and `speed2`. However, users that belong to a particular research group or project are will have a default Account like the following `aits`, `vidpro`, `gipsy`, `ai2`, `mpackir`, `cmos`, among others.

Directives are comments included at the beginning of a job script that set the shell and the options for the job scheduler.

The shebang directive is always the first line of a script. In your job script, this directive sets which shell your script's commands will run in. On "Speed", we recommend that your script use a shell from the `/encs/bin` directory.

To use `tcsh` shell, start your script with `#!/encs/bin/tcsh`, for `bash`, start with `#!/encs/bin/bash`

Directives that start with `#SBATCH` set the options for the cluster's SLURM job scheduler. The following provides an example of some essential directives:

```
#SBATCH --job-name=<jobname>        ## or -J. Give the job a name
#SBATCH --mail-type=<type>          ## set type of email notifications
#SBATCH --chdir=<directory>         ## or -D, set working directory for the job
#SBATCH --nodes=1                   ## or -N, node count required for the job
#SBATCH --ntasks=1                  ## or -n, number of tasks to be launched
#SBATCH --cpus-per-task=<corecount> ## or -c, core count requested, e.g. 8 cores
#SBATCH --mem=<memory>              ## assign memory for this job,
                                    ## e.g., 32G memory per node
```

Replace the following to adjust the job script for your project(s)

- `<jobname>` with a job name for the job. This name will be displayed in the job queue.
- `<directory>` with the fullpath to your job's working directory, e.g., where your code, source files and where the standard output files will be written to. By default, `--chdir` sets the current directory as the job's working directory.
- `<type>` with the type of e-mail notifications you wish to receive. Valid options are: NONE, BEGIN, END, FAIL, REQUEUE, ALL.
- `<corecount>` with the degree of multithreaded parallelism (i.e., cores) allocated to your job. Up to 32 by default.
- `<memory>` with the amount of memory, in GB, that you want to be allocated per node. Up to 500 depending on the node.
  Note: All jobs MUST set a value for the `--mem` option.

Example with short option equivalents:

```
#SBATCH -J myjob              ## Job's name set to 'myjob'
#SBATCH --mail-type=ALL       ## Receive all email type notifications
#SBATCH -D ./                 ## Use current directory as working directory
#SBATCH -N 1                  ## Node count required for the job
#SBATCH -n 1                  ## Number of tasks to be launched
#SBATCH -c 8                  ## Request 8 cores
#SBATCH --mem=32G             ## Allocate 32G memory per node
```

Tip: If you are unsure about memory footprints, err on assigning a generous memory space to your job, so that it does not get prematurely terminated. You can refine `--mem` values for future jobs by monitoring the size of a job's active memory space on `speed-submit` with:

```
sacct -j <jobID>
sstat -j <jobID>
```

This can be customized to show specific columns:

```
sacct -o jobid,maxvmsize,ntasks%7,tresusageouttot%25 -j <jobID>
sstat -o jobid,maxvmsize,ntasks%7,tresusageouttot%25 -j <jobID>
```

Memory-footprint efficiency values `seff` are also provided for completed jobs in the final email notification as "maxvmsize".

Jobs that request a low-memory footprint are more likely to load on a busy cluster.

Other essential options are `--time`, or `-t`, and `--account`, or `-A`.

- `--time=<time>` – is the estimate of wall clock time required for your job to run. As previously mentioned, the maximum is 7 days for batch and 24 hours for interactive jobs. Jobs with a smaller `time` value will have a higher priority and may result in your job being scheduled sooner.
- `--account=<name>` – specifies which Account, aka project or association, that the Speed resources your job uses should be attributed to. When moving from GE to SLURM users most users were assigned to Speed's two default accounts `speed1` and `speed2`. However, users that belong to a particular research group or project are will have a default Account like the following `aits`, `vidpro`, `gipsy`, `ai2`, `mpackir`, `cmos`, among others.

2.2.2 Working with Modules

After setting the directives in your job script, the next section typically involves loading the necessary software modules. The `module` command is used to manage the user environment, make sure to load all the modules your job depends on. You can check available modules with the module avail command. Loading the correct modules ensures that your environment is properly set up for execution.

To list for a particular program (`matlab`, for example):

```
module avail
module -t avail matlab  ## show the list for a particular program (e.g., matlab)
module -t avail m       ## show the list for all programs starting with 'm'
```

For example, insert the following in your script to load the `matlab/R2023a` module:

```
module load matlab/R2023a/default
```

Note: you can remove a module from active use by replacing `load` by `unload`.

To list loaded modules:

```
module list
```

To purge all software in your working environment:

```
module purge
```

2.2.3 User Scripting

The final part of the job script involves the commands that will be executed by the job. This section should include all necessary commands to set up and run the tasks your script is designed to perform. You can use any Linux command in this section, ranging from a simple executable call to a complex loop iterating through multiple commands.

Best Practice: prefix any compute-heavy step with `srun`. This ensures you gain proper insights on the execution of your job.

Each software program may have its own execution framework, as it's the script's author (e.g., you) responsibility to review the software's documentation to understand its requirements. Your script should be written to clearly specify the location of input and output files and the degree of parallelism needed.

Jobs that involve multiple interactions with data input and output files, should make use of `TMPDIR`, a scheduler-provided workspace nearly 1 TB in size. `TMPDIR` is created on the local disk of the compute node at the start of a job, offering faster I/O operations compared to shared storage (provided over NFS).

2.3 Sample Job Script

Here's a basic job script, `env.sh` shown in Figure 4. You can copy it from our GitHub repository.

```
#!/encs/bin/tcsh

#SBATCH --job-name=envs         ## Give the job a name
#SBATCH --mail-type=ALL         ## Receive all email type notifications
#SBATCH --chdir=./              ## Use currect directory as working directory
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1       ## Request 1 cpus
#SBATCH --mem=1G                ## Assign 1G memory per node

# Reset TMPDIR to a larger storage
mkdir -p /speed-scratch/$USER/tmp
setenv TMPDIR /speed-scratch/$USER/tmp

date
srun env
date

# EOF
```

Figure 4: Source code for `env.sh`

The first line is the shell declaration (also know as a shebang) and sets the shell to *tcsh*. The lines that begin with `#SBATCH` are directives for the scheduler.

- `-J` (or `--job-name`) sets *envs* as the job name.
- `--mail-type` sets the type of notifications.
- `--chdir` sets the working directory.
- `--nodes` specifies the number of required nodes.
- `--ntasks` specifies the number of tasks.

- `--cpus-per-task` requests 1 cpus.
- `--mem=` requests memory.

  Note: Jobs require the `--mem` option to be set either in the script or on the command line; **job submission will be rejected if it's missing.**

The script then:

1. Creates a directory.
2. Sets TMPDIR to a larger storage.
3. Prints current date.
4. Prints env variables.
5. Prints current date again.

The scheduler command, `sbatch`, is used to submit (non-interactive) jobs. From an ssh session on "speed-submit", submit this job with

```
sbatch ./env.sh
```

You will see, `Submitted batch job <JOB ID>`. The commands `squeue` and `sinfo` can be used to look at the queue and the status of the cluster

```
[serguei@speed-submit src] % squeue -l
Thu Oct 19 11:38:54 2023
JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
 2641        ps interact   b_user   RUNNING   19:16:09 1-00:00:00      1 speed-07
 2652        ps interact   a_user   RUNNING      41:40 1-00:00:00      1 speed-07
 2654        ps envs       serguei  RUNNING       0:01 7-00:00:00      1 speed-07

[serguei@speed-submit src] % sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
ps*          up 7-00:00:00      8  drng@ speed-[09-11,15-16,20-21,36]
ps*          up 7-00:00:00      3   drng speed-[38,42-43]
ps*          up 7-00:00:00      2  drain magic-node-[04,08]
ps*          up 7-00:00:00      4    mix magic-node-07,salus,speed-[07,37]
ps*          up 7-00:00:00      7  alloc magic-node-06,speed-[08,12,22-24,29]
ps*          up 7-00:00:00     13   idle magic-node-[05,09-10],speed-[19,30-35,39-41]
pg           up 7-00:00:00      1 drain* speed-05
pg           up 7-00:00:00      2    mix speed-[01,17]
pt           up 7-00:00:00      4   drng speed-[27,38,42-43]
pt           up 7-00:00:00      2    mix speed-[17,37]
pt           up 7-00:00:00      3   idle speed-[39-41]
pa           up 7-00:00:00      1   drng speed-27
pa           up 7-00:00:00      1    mix speed-01
pa           up 7-00:00:00      2   idle speed-[03,25]
cl           up 7-00:00:00      1 drain* speed-05
cl           up 7-00:00:00      4   drng speed-[27,38,42-43]
cl           up 7-00:00:00      3    mix speed-[01,17,37]
cl           up 7-00:00:00      6   idle speed-[03,19,25,39-41]
pc           up 7-00:00:00      1    mix salus
pm           up 7-00:00:00      2  drain magic-node-[04,08]
pm           up 7-00:00:00      1    mix magic-node-07
```

```
pm           up 7-00:00:00      1  alloc magic-node-06
pm           up 7-00:00:00      3   idle magic-node-[05,09-10]
pn           up 7-00:00:00      1  down* stellar
pn           up 7-00:00:00      2   idle matrix,nebulae
```

Once the job finishes, there will be a new file in the directory that the job was started from, with the syntax of, `slurm-<job id>.out`. This file represents the standard output (and error, if there is any) of the job in question. Important information is often written to this file.

2.4 Common Job Management Commands

Here is a summary of useful job management commands for handling various aspects of job submission and monitoring on the Speed cluster:

- Submitting a job:

  `sbatch -A <ACCOUNT> --mem=<MEMORY> -p <PARTITION> ./<myscript>.sh`

- Checking your job(s) status:

  `squeue -u <ENCSusername>`

- Displaying cluster status:

  `squeue`

  - Use `-A` for per account (e.g., `-A vidpro`, `-A aits`)
  - Use `-p` for per partition (e.g., `-p ps`, `-p pg`, `-p pt`), etc.

- Displaying job information:

  `squeue --job <job-ID>`

- Displaying individual job steps: (to see which step failed if you used `srun`)

  `squeue -las`

- Monitoring job and cluster status: (view `sinfo` and watch the queue for your job(s))

  `watch -n 1 "sinfo -Nel -pps,pt,pg,pa && squeue -la"`

- Canceling a job:

  `scancel <job-ID>`

- Holding a job:

  `scontrol hold <job-ID>`

- Releasing a job:

  `scontrol release <job-ID>`

- Getting job statistics: (including useful metrics like “maxvmem”)

  `sacct -j <job-ID>`

  `maxvmem` is one of the more useful stats that you can elect to display as a format option.

```
% sacct -j 2654
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
2654               envs          ps     speed1          1  COMPLETED      0:0
2654.batch        batch                speed1          1  COMPLETED      0:0
2654.extern      extern                speed1          1  COMPLETED      0:0
% sacct -j 2654 -o jobid,user,account,MaxVMSize,Reason%10,TRESUsageOutMax%30
JobID             User    Account  MaxVMSize     Reason        TRESUsageOutMax
------------ --------- ---------- ---------- ---------- ------------------------
2654           serguei     speed1                  None
2654.batch                 speed1    296840K             energy=0,fs/disk=1975
2654.extern                speed1    296312K              energy=0,fs/disk=343
```

See `man sacct` or `sacct -e` for details of the available formatting options. You can define your preferred default format in the `SACCT_FORMAT` environment variable in your `.cshrc` or `.bashrc` files.

- Displaying job efficiency: (including CPU and memory utilization)

  ```
  seff <job-ID>
  ```

  Don't execute it on `RUNNING` jobs (only on completed/finished jobs), else efficiency statistics may be misleading. If you define the following directive in your batch script, your GCS ENCS email address will receive an email with `seff`'s output when your job is finished.

  ```
  #SBATCH --mail-type=ALL
  ```

  Output example:

  ```
  Job ID: XXXXX
  Cluster: speed
  User/Group: user1/user1
  State: COMPLETED (exit code 0)
  Nodes: 1
  Cores per node: 4
  CPU Utilized: 00:04:29
  CPU Efficiency: 0.35% of 21:32:20 core-walltime
  Job Wall-clock time: 05:23:05
  Memory Utilized: 2.90 GB
  Memory Efficiency: 2.90% of 100.00 GB
  ```

2.5 Advanced `sbatch` Options

In addition to the basic sbatch options presented earlier, there are several advanced options that are generally useful:

- E-mail notifications: requests the scheduler to send an email when the job changes state.

  `--mail-type=<TYPE>`

  `<TYPE>` can be `ALL`, `BEGIN`, `END`, or `FAIL`.

  Mail is sent to the default address of `<ENCSusername>@encs.concordia.ca`

Which you can consult via `webmail.encs.concordia.ca` (use VPN from off-campus) unless a different address is supplied (see, `--mail-user`). The report sent when a job ends includes job runtime, as well as the maximum memory value hit (`maxvmem`).

To specify a different email address for notifications rather than the default, use

```
--mail-user email@domain.com
```

- Export environment variables used by the script:

```
--export=ALL
--export=NONE
--export=VARIABLES
```

- Job runtime: sets a job runtime of min or HH:MM:SS. Note that if you give a single number, that represents *minutes*, not hours. The set runtime should not exceed the default maximums of 24h for interactive jobs and 7 days for batch jobs.

```
-t <MINUTES> or DAYS-HH:MM:SS
```

- Job Dependencies: Runs the job only when the specified job `<job-ID>` finishes. This is useful for creating job chains where subsequent jobs depend on the completion of previous ones.

```
--depend=<state:job-ID>
```

Note: `sbatch` options can be specified during the job-submission command, and these *override* existing script options (if present). The syntax is

```
sbatch [options] path/to/script
```

but unlike in the script, the options are specified without the leading `#SBATCH` e.g.:

```
sbatch -J sub-test --chdir=./ --mem=1G ./env.sh
```

2.6 Array Jobs

Array jobs are those that start a batch job or a parallel job multiple times. Each iteration of the job array is called a task and receives a unique job ID. Array jobs are particularly useful for running a large number of similar tasks with slight variations.

To submit an array job (Only supported for batch jobs), use the `--array` option of the `sbatch` command as follows:

```
sbatch --array=n-m[:s] <script>
```

where

- `n`: indicates the start-id.
- `m`: indicates the max-id.
- `s`: indicates the step size.

Examples:

- Submit a job with 1 task where the task-id is 10.

```
sbatch --array=10 array.sh
```

- Submit a job with 10 tasks numbered consecutively from 1 to 10.

  ```
  sbatch --array=1-10 array.sh
  ```

- Submit a job with 5 tasks numbered consecutively with a step size of 3 (task-ids 3,6,9,12,15).

  ```
  sbatch --array=3-15:3 array.sh
  ```

- Submit a job with 50000 elements, where %a maps to the task-id between 1 and 50K.

  ```
  sbatch --array=1-50000 -N1 -i my_in_%a -o my_out_%a array.sh
  ```

Output files for Array Jobs: The default output and error-files are `slurm-job_id_task_id.out`. This means that Speed creates an output and an error-file for each task generated by the array-job, as well as one for the super-ordinate array-job. To alter this behavior use the `-o` and `-e` options of `sbatch`.

For more details about Array Job options, please review the manual pages for `sbatch` by executing the following at the command line on `speed-submit`.

```
man sbatch
```

2.7 Requesting Multiple Cores (i.e., Multithreading Jobs)

For jobs that can take advantage of multiple machine cores, you can request up to 32 cores (per job) in your script using the following options:

```
#SBATCH -n      #cores for processes>
#SBATCH -n 1
#SBATCH -c      #cores for threads of a single process
```

Both `sbatch` and `salloc` support `-n` on the command line, and it should always be used either in the script or on the command line as the default $n = 1$.

Important Considerations:

- Do not request more cores than you think will be useful, as larger-core jobs are more difficult to schedule.
- If you are running a program that scales out to the maximum single-machine core count available, please request 32 cores to avoid node oversubscription (i.e., overloading the CPUs).

Note:

- `--ntasks` or `--ntasks-per-node` (`-n`) refers to processes (usually the ones run with `srun`).
- `--cpus-per-task` (`-c`) corresponds to threads per process.

Some programs consider them equivalent, while others do not. For example, Fluent uses `--ntasks-per-node=8` and `--cpus-per-task=1`, whereas others may set `--cpus-per-task=8` and `--ntasks-per-node=1`. If one of these is not 1, some applications need to be configured to use `n * c` total cores.

Core count associated with a job appears under,"AllocCPUS", in the, `sacct -j <job-id>`, output.

```
[serguei@speed-submit src] % squeue -l
Thu Oct 19 20:32:32 2023
JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
2652        ps interact   a_user  RUNNING   9:35:18 1-00:00:00      1 speed-07
[serguei@speed-submit src] % sacct -j 2652
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
2652         interacti+         ps     speed1         20    RUNNING      0:0
2652.intera+ interacti+                speed1         20    RUNNING      0:0
2652.extern      extern                speed1         20    RUNNING      0:0
2652.0       gydra_pmi+                speed1         20  COMPLETED      0:0
2652.1       gydra_pmi+                speed1         20  COMPLETED      0:0
2652.2       gydra_pmi+                speed1         20     FAILED      7:0
2652.3       gydra_pmi+                speed1         20     FAILED      7:0
2652.4       gydra_pmi+                speed1         20  COMPLETED      0:0
2652.5       gydra_pmi+                speed1         20  COMPLETED      0:0
2652.6       gydra_pmi+                speed1         20  COMPLETED      0:0
2652.7       gydra_pmi+                speed1         20  COMPLETED      0:0
```

2.8 Interactive Jobs

Interactive job sessions allow you to interact with the system in real-time. These sessions are particularly useful for tasks such as testing, debugging, optimizing code, setting up environments, and other preparatory work before submitting batch jobs.

2.8.1 Command Line

To request an interactive job session, use the `salloc` command with appropriate options. This is similar to submitting a batch job but allows you to run shell commands interactively within the allocated resources. For example:

```
salloc -J interactive-test --mem=1G -p ps -n 8
```

Within the allocated `salloc` session, you can run shell commands as usual. It is recommended to use `srun` for compute-intensive steps within `salloc`. If you need a quick, short job just to compile something on a GPU node, you can use an interactive srun directly. For example, a 1-hour allocation:

For tcsh:

```
srun --pty -n 8 -p pg --gpus=1 --mem=1G -t 60 /encs/bin/tcsh
```

For bash:

```
srun --pty -n 8 -p pg --gpus=1 --mem=1G -t 60 /encs/bin/bash
```

2.8.2 Graphical Applications

To run graphical UI applications (e.g., MALTLAB, Abaqus CME, IDEs like PyCharm, VSCode, Eclipse, etc.) on Speed, you need to enable X11 forwarding from your client machine Speed then to the compute node. To do so, follow these steps:

1. Run an X server on your client machine:

- **Windows:** Use MobaXterm with X turned on, or Xming + PuTTY with X11 forwarding, or XOrg under Cygwin
- **macOS:** Use XQuarz – use its `xterm` and `ssh -X`
- **Linux:** Use `ssh -X speed.encs.concordia.ca`

For more details, see How do I remotely launch X(Graphical) applications?

2. Verify that X11 forwarding is enabled by printing the `DISPLAY` variable:

```
echo $DISPLAY
```

3. Start an interactive session with X11 forwarding enabled (Use the `--x11` with `salloc` or `srun`), for example:

```
salloc -p ps --x11=first --mem=4G -t 0-06:00
```

4. Once landed on a compute node, verify `DISPLAY` again.

5. Set the `XDG_RUNTIME_DIR` variable to a directory in your `speed-scratch` space:

```
mkdir -p /speed-scratch/$USER/run-dir
setenv XDG_RUNTIME_DIR /speed-scratch/$USER/run-dir
```

6. Launch your graphical application:

```
module load matlab/R2023a/default
matlab
```

Note: with X11 forwarding the graphical rendering is happening on your client machine! That is you are not using GPUs on Speed to render graphics, instead all graphical information is forwarded from Speed to your desktop or laptop over X11, which in turn renders it using its own graphics card. Thus, for GPU rendering jobs either keep them non-interactive or use VirtualGL.

Here's an example of starting PyCharm (see Figure 5). **Note:** If using VSCode, it's currently only supported with the `--no-sandbox` option.

TCSH version:

```
ssh -X speed (XQuartz xterm, PuTTY or MobaXterm have X11 forwarding too)
[speed-submit] [/home/c/carlos] > echo $DISPLAY
localhost:14.0
[speed-submit] [/home/c/carlos] > cd /speed-scratch/$USER
[speed-submit] [/speed-scratch/carlos] > echo $DISPLAY
localhost:13.0
[speed-submit] [/speed-scratch/carlos] > salloc -pps --x11=first --mem=4Gb -t 0-06:00
[speed-07] [/speed-scratch/carlos] > echo $DISPLAY
localhost:42.0
[speed-07] [/speed-scratch/carlos] > hostname
speed-07.encs.concordia.ca
[speed-07] [/speed-scratch/carlos] > setenv XDG_RUNTIME_DIR /speed-scratch/$USER/run-dir
[speed-07] [/speed-scratch/carlos] > /speed-scratch/nag-public/bin/pycharm.sh
```

BASH version:

```
bash-3.2$ ssh -X speed (XQuartz xterm, PuTTY or MobaXterm have X11 forwarding too)
serguei@speed's password:
```

```
[serguei@speed-submit ~] % echo $DISPLAY
localhost:14.0
[serguei@speed-submit ~] % salloc -p ps --x11=first --mem=4Gb -t 0-06:00
bash-4.4$ echo $DISPLAY
localhost:77.0
bash-4.4$ hostname
speed-01.encs.concordia.ca
bash-4.4$ export XDG_RUNTIME_DIR=/speed-scratch/$USER/run-dir
bash-4.4$ /speed-scratch/nag-public/bin/pycharm.sh
```

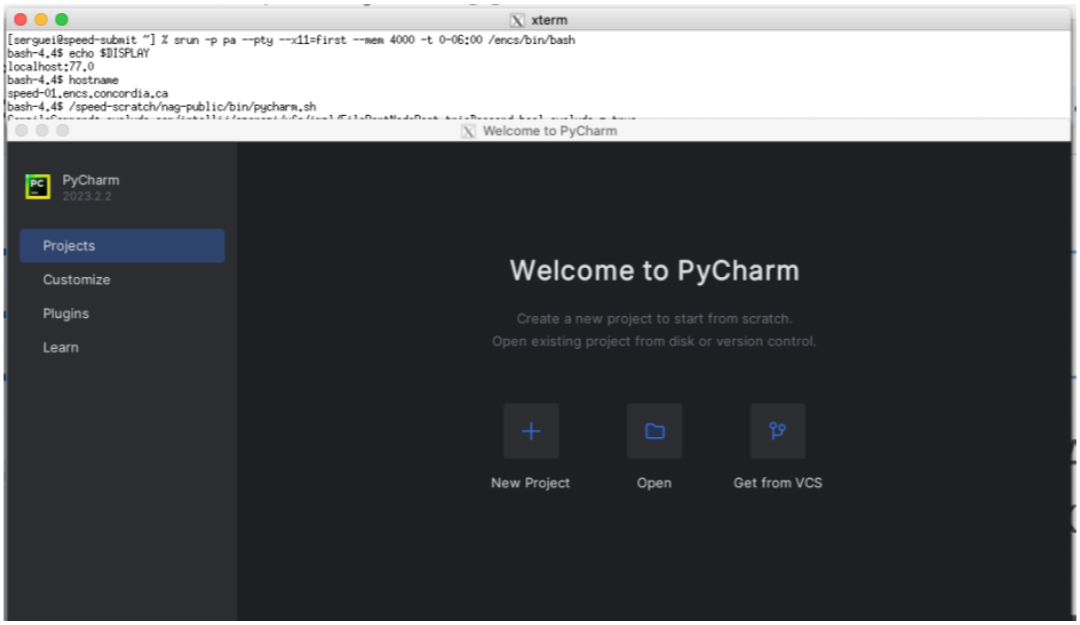

Figure 5: Launching PyCharm on a Speed Node

2.8.3 Jupyter Notebooks

2.8.3.1 Jupyter Notebook in Singularity To run Jupyter Notebooks using Singularity (more on Singularity see Section 2.16), follow these steps:

1. Connect to Speed, e.g. interactively, using `salloc`
2. Load Singularity module `module load singularity/3.10.4/default`
3. Execute this Singularity command on a single line or save it in a shell script from our GitHub repo where you could easily invoke it.

   ```
   srun singularity exec -B $PWD\:/speed-pwd,/speed-scratch/$USER\:/my-speed-scratch,/nettemp \
   --env SHELL=/bin/bash --nv /speed-scratch/nag-public/openiss-cuda-conda-jupyter.sif \
   /bin/bash -c '/opt/conda/bin/jupyter notebook --no-browser --notebook-dir=/speed-pwd \
   --ip="*" --port=8888 --allow-root'
   ```

4. In a new terminal window, create an `ssh` tunnel between your computer and the node (`speed-XX`) where Jupyter is running (using `speed-submit` as a "jump server", see, e.g., in PuTTY, in Figure 6 and Figure 7)

```
ssh -L 8888:speed-XX:8888 <ENCS-username>@speed-submit.encs.concordia.ca
```

Don't close the tunnel after establishing.

5. Open a browser, and copy your Jupyter's token (it's printed to you in the terminal) and paste it in the browser's URL field. In our case, the URL is:

```
http://localhost:8888/?token=5a52e6c0c7dfc111008a803e5303371ed0462d3d547ac3fb
```

6. Access the Jupyter Notebook interface in your browser.

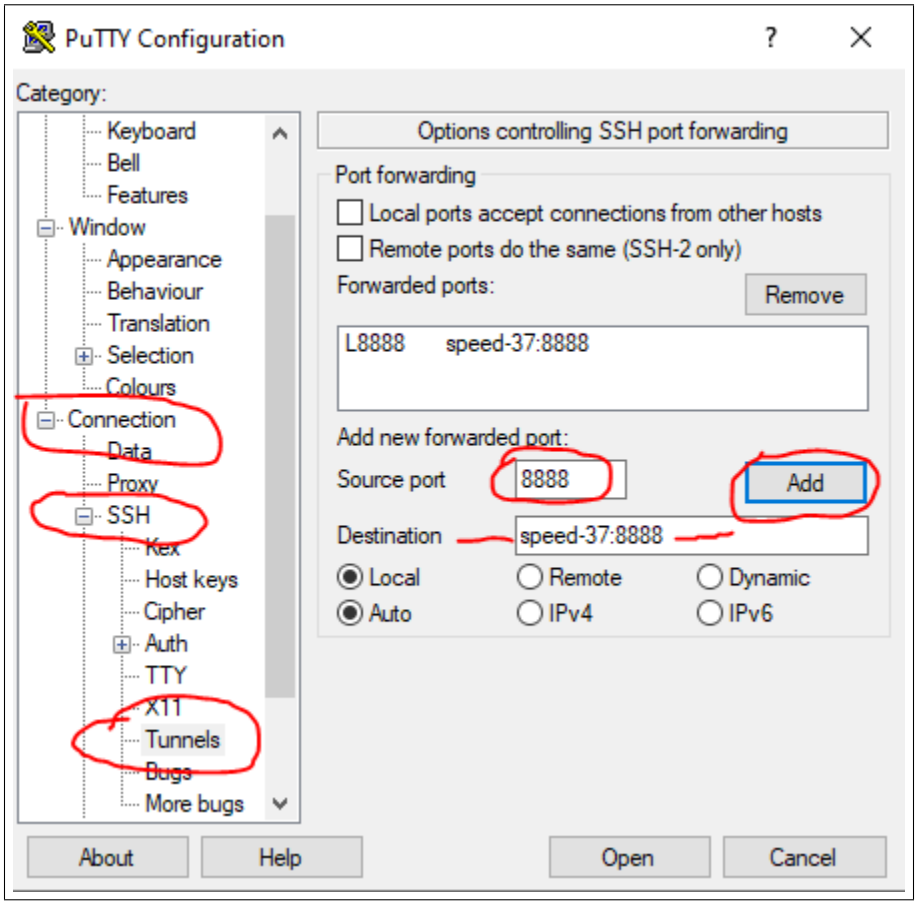

Figure 6: SSH tunnel configuration 1

Another sample is the OpenISS-derived containers with Conda and Jupyter, see Section 2.15.4 for details.

2.8.3.2 Jupyter Notebook in Conda For setting up Jupyter Labs with Conda and Pytorch, follow these steps:

Environment preparation: (only once, takes some time to run to install all required dependencies)

1. Start an interactive session, and navigate to your `speed-scratch` directory:

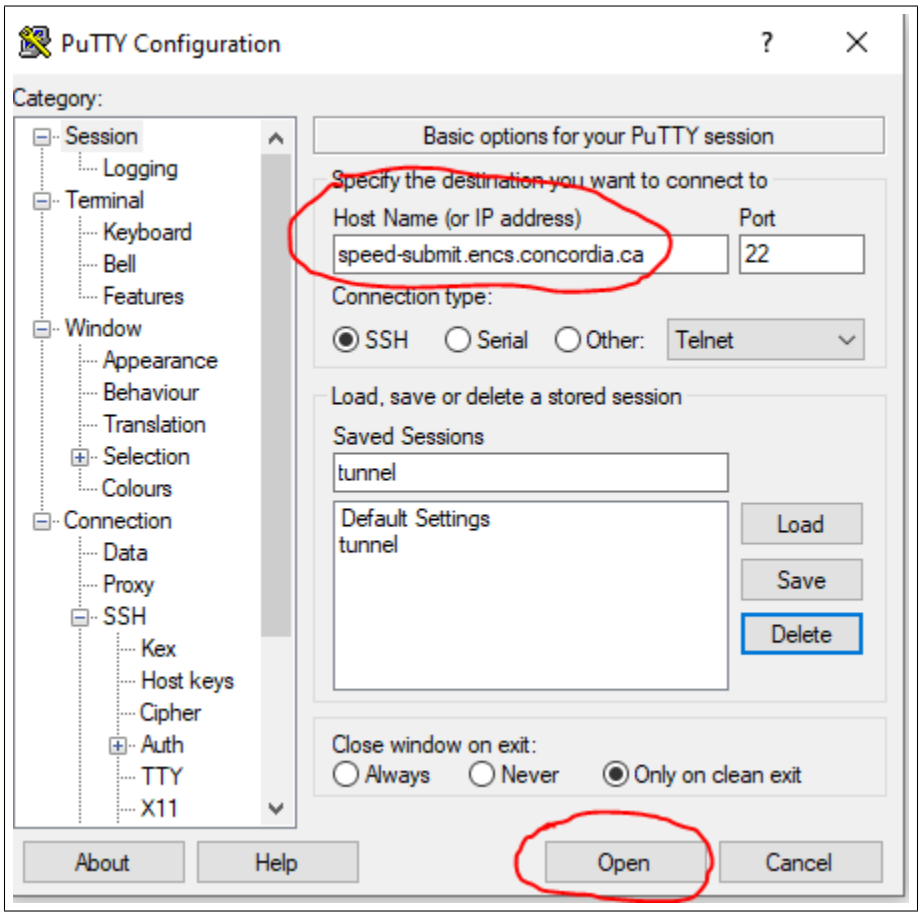

Figure 7: SSH tunnel configuration 2

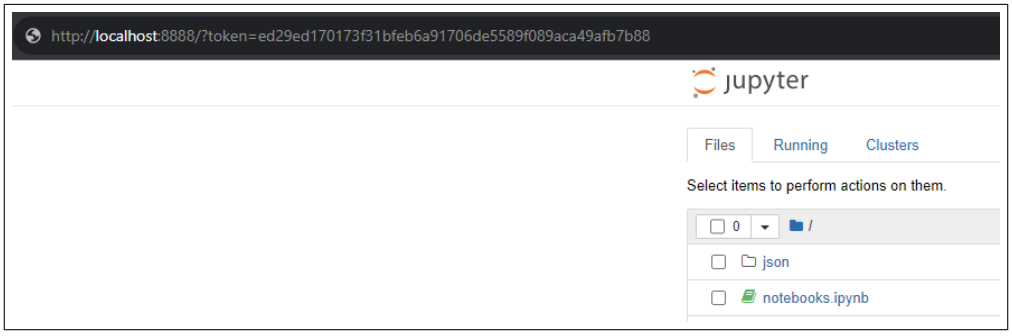

Figure 8: Jupyter running on a Speed node

```
salloc --mem=20G --gpus=1
cd /speed-scratch/$USER
```

2. Load and initilize the environment

```
module load anaconda/2023.03/default
conda init tcsh
source ~/.tcshrc
```

3. Set up Conda environment by runnung `setup-conda.sh` (on the compute node salloc brought you to, not on speed-submit) as shown in Figure 9

```
./setup_conda.sh
```

```
#!/encs/bin/tcsh

mkdir -p /speed-scratch/$USER/Jupyter
module load anaconda3/2023.03/default
setenv TMPDIR /speed-scratch/$USER/tmp
setenv TMP /speed-scratch/$USER/tmp
setenv CONDA_PKGS_DIRS /speed-scratch/$USER/Jupyter/pkgs
conda create -p /speed-scratch/$USER/Jupyter/jupyter-env -y
conda activate /speed-scratch/$USER/Jupyter/jupyter-env
conda install -c conda-forge jupyterlab -y
pip3 install --quiet torch --index-url https://download.pytorch.org/whl/cu118
exit
```

Figure 9: Source code for `setup_conda.sh`

The script will:

- create a Jupyter directory change Jupyter to any name of your choice in the script
- set environment variables
- create a conda environment named jupyter-env
- install JupyterLabs and pytorch
- exit the interactive session

Launching Jupyter Labs instance from speed-submit:

1. Run the `start_jupyterlab.sh` script each time you need to launch JupyterLab from the submit node The script will:
   - allocate resources for your job on a compute node
   - start jupyter server by running `run_jupyterlab.sh`
   - print the ssh command that you can use to connect to the compute node runnung the jupyter notebook (this is done in a new terminal)
   - print the token/link to the jupyter server to paste in a web browser (starting with http://127.0.0.1/...)
2. Open a browser, and copy your Jupyter's token and paste it in the browser's URL field.

2.8.3.3 Jupyter Notebook in Python venv

This is an example of Jupyter Labs running in a Python Virtual environment on Speed.

Note: Use of Python virtual environments is preferred over Conda at Alliance Canada clusters. If you prefer to make jobs that are more compatible between Speed and Alliance clusters, use Python `venv`s. See `https://docs.alliancecan.ca/wiki/Anaconda/en` and `https://docs.alliancecan.ca/wiki/JupyterNotebook`.

- Environment preparation: for the FIRST time only:
  1. Go to your speed-scratch directory: `cd /speed-scratch/$USER`
  2. Open an interactive session: `salloc --mem=50G --gpus=1 --constraint=el9`
  3. Create a Python `venv` and install `jupyterlab+pytorch`

```
module load python/3.11.5/default
setenv TMPDIR /speed-scratch/$USER/tmp
setenv TMP /speed-scratch/$USER/tmp
setenv PIP_CACHE_DIR /speed-scratch/$USER/tmp/cache
python -m venv /speed-scratch/$USER/tmp/jupyter-venv
source /speed-scratch/$USER/tmp/jupyter-venv/bin/activate.csh
pip install jupyterlab
pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
exit
```

- Running Jupyter, from **speed-submit**:
  1. Open an interactive session: `salloc --mem=50G --gpus=1 --constraint=el9`

```
cd /speed-scratch/$USER
module load python/3.11.5/default
setenv PIP_CACHE_DIR /speed-scratch/$USER/tmp/cache
source /speed-scratch/$USER/tmp/jupyter-venv/bin/activate.csh
jupyter lab --no-browser --notebook-dir=$PWD --ip="0.0.0.0" --port=8888 --port-retries=50
```

  2. Verify which port the system has assigned to Jupyter: `http://localhost:XXXX/lab?token=`
  3. SSH Tunnel creation: similar to Jupyter in Singularity, see Section 2.8.3.1
  4. Open a browser and type: `localhost:XXXX` (using the port assigned)

2.8.4 Visual Studio Code

This is an example of running VScode, it's similar to Jupyter notebooks, but it doesn't use containers. **Note:** this a Web-based version; there exists the local (workstation) – remote (speed-node) client-server version too, but it is for advanced users and is out of scope here (so no support, use it at your own risk).

- Environment preparation: for the FIRST time:
  1. Go to your speed-scratch directory: `cd /speed-scratch/$USER`
  2. Create a vscode directory: `mkdir vscode`
  3. Create another directory: `mkdir -p /speed-scratch/$USER/run-user`
- Running VScode

1. Go to your vscode directory: `cd /speed-scratch/$USER/vscode`
2. Open interactive session: `salloc --mem=10Gb --constraint=el9`
3. Set environment variable: `setenv XDG_RUNTIME_DIR /speed-scratch/$USER/run-user`
4. Run VScode, change the port if needed.

```
/speed-scratch/nag-public/code-server-4.22.1/bin/code-server --user-data-dir=$PWD\/projects \
--config=$PWD\/home/.config/code-server/config.yaml --bind-addr="0.0.0.0:8080" $PWD\/projects
```

5. SSH Tunnel creation: similar to Jupyter, see Section 2.8.3.1
6. Open a browser and type: `localhost:8080`
7. If the browser asks for a password, consult:

```
cat /speed-scratch/$USER/vscode/home/.config/code-server/config.yaml
```

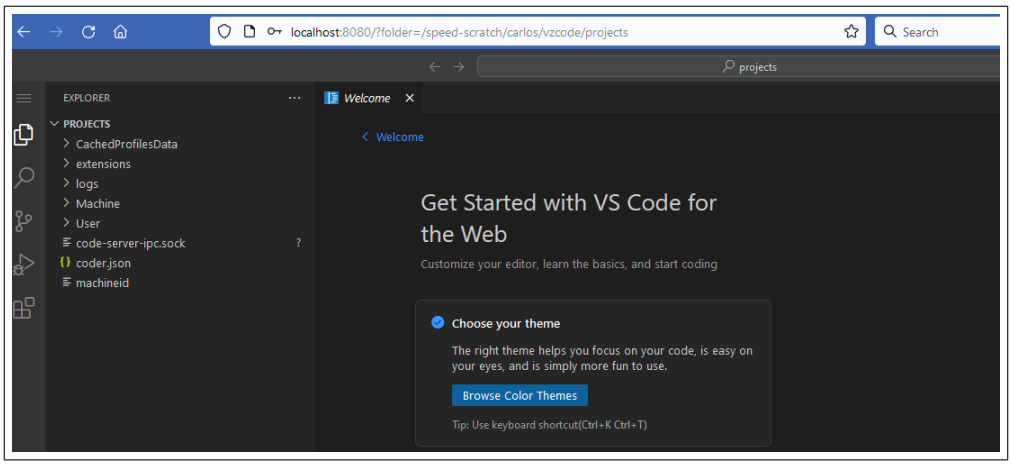

Figure 10: VScode running on a Speed node

2.9 Scheduler Environment Variables

The scheduler provides several environment variables that can be useful in your job scripts. These variables can be accessed within the job using commands like `env` or `printenv`. Many of these variables start with the prefix `SLURM`.

Here are some of the most useful environment variables:

- `$TMPDIR` (and `$SLURM_TMPDIR`): This is the path to the job's temporary space on the node. It *only* exists for the duration of the job. If you need the data from this temporary space, ensure you copy it before the job terminates.
- `$SLURM_SUBMIT_DIR`: The path to the job's working directory (likely an NFS-mounted path). If, `--chdir`, was stipulated, that path is taken; if not, the path defaults to your home directory.

- `$SLURM_JOBID`: This variable holds the current job's ID, which is useful for job manipulation and reporting within the job's process.
- `$SLURM_NTASKS`: the number of cores requested for the job. This variable can be used in place of hardcoded thread-request declarations, e.g., for Fluent or similar.
- `$SLURM_JOB_NODELIST`: This lists the nodes participating in your job.
- `$SLURM_ARRAY_TASK_ID`: For array jobs, this variable represents the task ID (refer to Section 2.6 for more details on array jobs).

For a more comprehensive list of environment variables, refer to the SLURM documentation for Input Environment Variables and Output Environment Variables.

An example script that utilizes some of these environment variables is in Figure 11.

```
#!/encs/bin/tcsh

#SBATCH --job-name=tmpdir      ## Give the job a name
#SBATCH --mail-type=ALL        ## Receive all email type notifications
#SBATCH --chdir=./             ## Use currect directory as working directory
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8      ## Request 8 cores
#SBATCH --mem=32G              ## Assign 32G memory per node

cd $TMPDIR
mkdir input
rsync -av $SLURM_SUBMIT_DIR/references/ input/
mkdir results
srun STAR --inFiles $TMPDIR/input --parallel $SRUN_CPUS_PER_TASK --outFiles $TMPDIR/results
rsync -av $TMPDIR/results/ $SLURM_SUBMIT_DIR/processed/
```

Figure 11: Source code for `tmpdir.sh`

2.10 SSH Keys for MPI

Some programs, such as Fluent, utilize MPI (Message Passing Interface) for parallel processing. MPI requires 'passwordless login', which is achieved through SSH keys. Here are the steps to set up SSH keys for MPI:

- Navigate to the `.ssh` directory

  `cd ~/.ssh`

- Generate a new SSH key pair (Accept the default location and leave the passphrase blank)

  `ssh-keygen -t ed25519`

- Authorize the Public Key:

  `cat id_ed25519.pub >> authorized_keys`

  If the `authorized_keys` file does not exist, use

  `cat id_ed25519.pub > authorized_keys`

- Set permissions: ensure the correct permissions are set for the 'authorized_keys' file and your home directory (most users will already have these permissions by default):

```
chmod 600 ~/.ssh/authorized_keys
chmod 700 ~
```

2.11 Creating Virtual Environments

The following documentation is specific to **Speed**, other clusters may have their own requirements.

Virtual environments are typically created using Conda or Python. Another option is Singularity (detailed in Section 2.16). These environments are usually created once during an interactive session before submitting a batch job to the scheduler.

The job script submitted to the scheduler should:

1. Activate the virtual environment.
2. Use the virtual environment.
3. Deactivate the virtual environment at the end of the job.

2.11.1 Anaconda

To create an Anaconda environment, follow these steps:

1. Request an interactive session

   ```
   salloc -p pg --gpus=1
   ```

2. Load the Anaconda module and create your Anaconda environment in your speed-scratch directory by using the `--prefix` option (without this o tion, the environment will be created in your home directory by default).

   ```
   module load anaconda3/2023.03/default
   conda create --prefix /speed-scratch/$USER/myconda
   ```

3. List environments (to view your conda environment)

   ```
   conda info --envs
   # conda environments:
   #
   base                  *  /encs/pkg/anaconda3-2023.03/root
                            /speed-scratch/a_user/myconda
   ```

4. Activate the environment

   ```
   conda activate /speed-scratch/$USER/myconda
   ```

5. Add `pip` to your environment (this will install `pip` and `pip`'s dependencies, including `python`, into the environment.)

   ```
   conda install pip
   ```

A consolidated example using Conda:

```
salloc --mem=10G --gpus=1 -p pg -A <slurm account name>
mkdir -p /speed-scratch/$USER
cd /speed-scratch/$USER
```

```
module load anaconda3/2023.03/default
conda create -p /speed-scratch/$USER/pytorch-env
conda activate /speed-scratch/$USER/pytorch-env
conda install python=3.11.0
pip3 install torch torchvision torchaudio --index-url \
https://download.pytorch.org/whl/cu117
....
conda deactivate
exit # end the salloc session
```

If you encounter **no space left error** while creating Conda environments, please refer to Appendix B.3. Likely you forgot `--prefix` or environment variables below.

Important Note: `pip` (and `pip3`) are package installers for Python. When you use `pip install`, it installs packages from the Python Package Index (PyPI), whereas, `conda install` installs packages from Anaconda's repository.

2.11.1.1 Conda Env without `--prefix`

If you don't want to use the `--prefix` option every time you create a new environment and do not want to use the default home directory, you can create a new directory and set the following variables to point to the newly created directory, e.g.:

```
mkdir -p /speed-scratch/$USER/conda
setenv CONDA_ENVS_PATH /speed-scratch/$USER/conda
setenv CONDA_PKGS_DIRS /speed-scratch/$USER/conda/pkg
```

If you want to make these changes permanent, add the variables to your `.tcshrc` or `.bashrc` (depending on the default shell you are using).

2.11.2 Python

Setting up a Python virtual environment is straightforward. Here's an example that use a Python virtual environment:

```
salloc --mem=10G --gpus=1 -p pg -A <slurm account name>
mkdir -p /speed-scratch/$USER
cd /speed-scratch/$USER
module load python/3.9.1/default
mkdir -p /speed-scratch/$USER/tmp
setenv TMPDIR /speed-scratch/$USER/tmp
setenv TMP /speed-scratch/$USER/tmp
python -m venv $TMPDIR/testenv (testenv=name of the virtualEnv)
source /speed-scratch/$USER/tmp/testenv/bin/activate.csh
pip install <modules>
deactivate
exit
```

See, e.g., `gurobi-with-python.sh`

Important Note: our partition `ps` is used for CPU jobs, while `pg`, `pt`, and `cl` are used for GPU jobs. You do not need to use `--gpus` when preparing environments for CPU jobs.

Note: Python enviornments are also preferred over Conda in some clusters, see a note in Section 2.8.3.3.

2.12 Example Job Script: Fluent

```
#!/encs/bin/tcsh

#SBATCH --job-name=flu10000     ## Give the job a name
#SBATCH --mail-type=ALL         ## Receive all email type notifications
#SBATCH --chdir=./              ## Use currect directory as working directory
#SBATCH --nodes=1               ## Number of nodes to run on
#SBATCH --ntasks-per-node=32    ## Number of cores
#SBATCH --cpus-per-task=1       ## Number of MPI threads
#SBATCH --mem=160G              ## Assign 160G memory per node

date

module avail ansys

module load ansys/2023R2/default
cd $TMPDIR

set FLUENTNODES = "`scontrol␣show␣hostnames`"
set FLUENTNODES = `echo $FLUENTNODES | tr ' ' ','`

date

srun fluent 3ddp \
        -g -t$SLURM_NTASKS \
        -g-cnf=$FLUENTNODES \
        -i $SLURM_SUBMIT_DIR/fluentdata/info.jou > call.txt

date

srun rsync -av $TMPDIR/ $SLURM_SUBMIT_DIR/fluentparallel/

date
```

Figure 12: Source code for `fluent.sh`

The job script in Figure 12 runs Fluent in parallel over 32 cores. Notable aspects of this script include requesting e-mail notifications (`--mail-type`), defining the parallel environment for Fluent with `-t$SLURM_NTASKS` and `-g-cnf=$FLUENTNODES`, and setting `$TMPDIR` as the in-job location for the "moment" `rfile.out` file. The script also copies everything from `$TMPDIR` to a directory in the user's NFS-mounted home after the job completes. Job progress can be monitored by examining the standard-out file (e.g., `slurm-249.out`), and/or by examining the "moment" file in TMPDIR (usually `/disk/nobackup/<yourjob>` (it starts with your job-ID)) on the node running the job. Be cautious with journal-file paths.

2.13 Example Job: EfficientDet

The following steps describe how to create an EfficientDet environment on Speed, as submitted by a member of Dr. Amer's research group:

- Navigate to your `speed-scratch` directory:

  ```
  cd /speed-scratch/$USER
  ```

- Load Python module

  ```
  module load python/3.8.3
  ```

- Create and activate the virtual environment

  ```
  python3 -m venv <env_name>
  source <env_name>/bin/activate.csh
  ```

- Install DL packages for EfficientDet

  ```
  pip install tensorflow==2.7.0
  pip install lxml>=4.6.1
  pip install absl-py>=0.10.0
  pip install matplotlib>=3.0.3
  pip install numpy>=1.19.4
  pip install Pillow>=6.0.0
  pip install PyYAML>=5.1
  pip install six>=1.15.0
  pip install tensorflow-addons>=0.12
  pip install tensorflow-hub>=0.11
  pip install neural-structured-learning>=1.3.1
  pip install tensorflow-model-optimization>=0.5
  pip install Cython>=0.29.13
  pip install git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI
  ```

2.14 Java Jobs

Jobs that call Java have a memory overhead, which needs to be taken into account when assigning a value to `--mem`. Even the most basic Java call, such as `Java -Xmx1G -version`, will need to have, `--mem=5G`, with the 4 GB difference representing the memory overhead. **Note** that this memory overhead grows proportionally with the value of `-Xmx`. For example,

- When `-Xmx` has a value of 100G, `--mem` has to be at least 106G.
- For `-Xmx` of 200G, `--mem` has to be at least 211G.
- For `-Xmx` of 300G, `--mem` has to be at least 314G.

2.15 Scheduling on the GPU Nodes

Speed has various GPU types in various subclusters of its nodes.

- `speed-05` and `speed-17`: The primary SPEED1 cluster has two GPU nodes, each with six Tesla (CUDA-compatible) P6 cards. Each card has 2048 cores and 16GB of RAM. Note that the P6 is mainly a single-precision card, so unless you need GPU double precision, double-precision calculations will be faster on a CPU node.
- `speed-01`: This `vidpro` node (see Figure 2, contact Dr. Maria Amer) is identical to 05 and 17 in its GPU configuration, but managed by the priority for the vidpro group, that is a `pg` job scheduled there is a subject for preemption.
- `speed-03`, `speed-25`, `speed-25`: These `vidpro` nodes feature NVIDIA V100 cards with 32GB of RAM. Like `speed-01`, the priority is of the vidpro group, who purchased the nodes, and others' jobs are a subject from preemption within `pg`, `pt`, and `cl` partitions.
- `speed-37` – `speed-43`: SPEED2 nodes, the main backbone of the teaching partition `pt`, have 4x A100 80GB GPUs each, partitioned into average 4x MIGs of 20GB each, with exceptions.
- `nebulae`: A member of the Nebular subcluster (contact Dr. Jun Yan), has 2x 48GB RTX Ada 6000 cards. This node is in the `pn` partition.
- `speed-19`: Has an AMD GPU, Tonga, 16GB of GPU ram. This node along with the majority of the NVIDIA GPU nodes are in the `cl` partition (with restrictions) to run OpenCL, Vulkan, and HIP jobs.

Job scripts for the GPU queues differ in that they need these statements, which attach either a single GPU or more GPUs to the job with the appropriate partition:

```
#SBATCH --gpus=[1|x]
#SBATCH -p [pg|pt|cl|pa]
```

The default max quota for x is 4.

Once your job script is ready, submit it to the GPU partition (queue) with:

```
sbatch --mem=<MEMORY> -p pg ./<myscript>.sh
```

`--mem` and `-p` can reside in the script.

You can query `nvidia-smi` on the node **running your job** with:

```
ssh <ENCSusername>@speed-[01|03|05|17|25|27|37-43]|nebulae nvidia-smi
```

The status of the GPU queues can be queried e.g. with:

```
sinfo -p pg --long --Node
sinfo -p pt --long --Node
sinfo -p cl --long --Node
sinfo -p pa --long --Node
sinfo -p pn --long --Node
```

You can query `rocm-smi` on the AMD GPU node running your job with:

```
ssh <ENCSusername>@speed-19 rocm-smi
```

Important note for TensorFlow and PyTorch users: if you are planning to run TensorFlow and/or PyTorch multi-GPU jobs, please **do not use** the `tf.distribute` and/or `torch.nn.DataParallel` functions on **speed-01, speed-05, or speed-17**, as they will crash

the compute node (100% certainty). This appears to be a defect in the current hardware architecture. The workaround is to either manually effect GPU parallelisation (see Section 2.15.1) (TensorFlow provides an example on how to do this), or to run on a single GPU, which is now the default for those nodes.

Important: Users without permission to use the GPU nodes can submit jobs to the various GPU partitions, but those jobs will hang and never run. Their availability can be seen with:

```
[serguei@speed-submit src] % sinfo -p pg --long --Node
Thu Oct 19 22:31:04 2023
NODELIST   NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE REASON
speed-05       1        pg        idle 32     2:16:1 515490        0      1    gpu16 none
speed-17       1        pg     drained 32     2:16:1 515490        0      1    gpu16 UGE
speed-25       1        pg        idle 32     2:16:1 257458        0      1    gpu32 none
speed-27       1        pg        idle 32     2:16:1 257458        0      1    gpu32 none
[serguei@speed-submit src] % sinfo -p pt --long --Node
Thu Oct 19 22:32:39 2023
NODELIST   NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE REASON
speed-37       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-38       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-39       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-40       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-41       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-42       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-43       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
```

To specifically request a GPU node, add, `--gpus=[#GPUs]`, to your `sbatch` statement/script or `salloc` statement request. For example:

```
sbatch -t 10 --mem=1G --gpus=1 -p pg ./tcsh.sh
```

The request can be further specified to a specific node using `-w` or a GPU type or feature.

```
[serguei@speed-submit src] % squeue -p pg -o "%15N %.6D %7P %.11T %.4c %.8z %.6m %.8d %.6w %.8f %20G %20E"
NODELIST         NODES PARTITI       STATE MIN_    S:C:T MIN_ME MIN_TMP_  WCKEY FEATURES GROUP DEPENDENCY
speed-05             1 pg          RUNNING    1    *:*:*     1G        0 (null)   (null) 11929     (null)
[serguei@speed-submit src] % sinfo -p pg -o "%15N %.6D %7P %.11T %.4c %.8z %.6m %.8d %.6w %.8f %20G %20E"
NODELIST         NODES PARTITI       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE GRES      REASON
speed-17             1 pg          drained   32   2:16:1 515490        0      1    gpu16 gpu:6        UGE
speed-05             1 pg            mixed   32   2:16:1 515490        0      1    gpu16 gpu:6       none
speed-[25,27]        2 pg             idle   32   2:16:1 257458        0      1    gpu32 gpu:2       none
```

2.15.1 P6 on Multi-GPU, Multi-Node

As described earlier, P6 cards are not compatible with `Distribute` and `DataParallel` functions (**PyTorch**, **Tensorflow**) when running on multiple GPUs. One workaround is to run the job in Multi-node, single GPU per node (this applies to P6 nodes: speed-05, speed-17, speed-01):

```
#SBATCH --nodes=2
#SBATCH --gpus-per-node=1
```

An example script for training on multiple nodes with multiple GPUs is provided in pytorch-multinode-multigpu.sh illustrates a job for training on Multi-Nodes, Multi-GPUs

2.15.2 CUDA

When calling **CUDA** within job scripts, it is important to link to the desired the desired **CUDA** libraries and set the runtime link path to the same libraries. For example, to use the `cuda-11.5` libraries, specify the following in your `Makefile`.

```
-L/encs/pkg/cuda-11.5/root/lib64 -Wl,-rpath,/encs/pkg/cuda-11.5/root/lib64
```

In your job script, specify the version of `GCC` to use prior to calling CUDA:

```
module load gcc/9.3
```

2.15.3 Special Notes for Sending CUDA Jobs to the GPU Queues

Interactive jobs (Section 2.8) must be submitted to the GPU partition to compile and link. Several versions of CUDA are installed in:

```
/encs/pkg/cuda-11.5/root/
/encs/pkg/cuda-10.2/root/
/encs/pkg/cuda-9.2/root
```

For CUDA to compile properly for the GPU partition, edit your `Makefile` replacing `usrlocalcuda` with one of the above.

2.15.4 OpenISS Examples

These examples represent more comprehensive research-like jobs for computer vision and other tasks with longer runtime (subject to the number of epochs and other parameters). They derive from the actual research works of students and their theses and require the use of CUDA and GPUs. These examples are available as "native" jobs on Speed and as Singularity containers.

Examples include:

2.15.4.1 OpenISS and REID A computer-vision-based person re-identification (e.g., motion capture-based tracking for stage performance) part of the OpenISS project by Haotao Lai [12] using TensorFlow and Keras. The script is available here: openiss-reid-speed.sh. The fork of the original repo [14] adjusted to run on Speed is available here: openiss-reid-tfk. Detailed instructions on how to run it on Speed are in the README: `https://github.com/NAG-DevOps/speed-hpc/tree/master/src#openiss-reid-tfk`

2.15.4.2 OpenISS and YOLOv3 The related code using YOLOv3 framework is in the the fork of the original repo [13] adjusted to to run on Speed is available here: openiss-yolov3.

Example job scripts can run on both CPUs and GPUs, as well as interactively using TensorFlow:

- Interactive mode: openiss-yolo-interactive.sh
- CPU-based job: openiss-yolo-cpu.sh
- GPU-based job: openiss-yolo-gpu.sh

Detailed instructions on how to run these on Speed are in the README: `https://github.com/NAG-DevOps/speed-hpc/tree/master/src#openiss-yolov3`

2.16 Singularity Containers

Singularity is a container platform designed to execute applications in a portable, reproducible, and secure manner. Unlike Docker, Singularity does not require root privileges, making it more suitable for HPC environments. If the `/encs` software tree does not have the required software available, another option is to run Singularity containers. We run EL7 and EL9 flavors of Linux, and if some projects require Ubuntu or other distributions, it is possible to run that software as a container, including those converted from Docker. The currently recommended version of Singularity is `singularity/3.10.4/default`.

The example lambdal-singularity.sh showcases an immediate use of a container built for the Ubuntu-based LambdaLabs software stack, originally built as a Docker image then pulled in as a Singularity container. The source material used for the docker image was our fork of their official repository: `https://github.com/NAG-DevOps/lambda-stack-dockerfiles`.

Note: If you make your own containers or pull from DockerHub, use your `/speed-scratch/$USER` directory, as these images may easily consume gigabytes of space in your home directory, quickly exhausting your quota.

Tip: To check your quota and find big files, see Section B.3 and ENCS Data Storage.

We have also built equivalent OpenISS (Section 2.15.4) containers from their Docker counterparts for teaching and research purposes [16]. The images from `https://github.com/NAG-DevOps/openiss-dockerfiles` and their DockerHub equivalents `https://hub.docker.com/u/openiss` can be found in `/speed-scratch/nag-public` with a '`.sif`' extension. Some can be run in both batch and interactive modes, covering basics with CUDA, OpenGL rendering, and computer vision tasks. Examples include Jupyter notebooks with Conda support.

```
/speed-scratch/nag-public:
openiss-cuda-conda-jupyter.sif
openiss-cuda-devicequery.sif
openiss-opengl-base.sif
openiss-opengl-cubes.sif
openiss-opengl-triangle.sif
openiss-reid.sif
openiss-xeyes.sif
```

This section introduces working with Singularity, its containers, and what can and cannot be done with Singularity on the ENCS infrastructure. For comprehensive documentation, refer to the authors' guide: `https://www.sylabs.io/docs/`.

Singularity containers are either built from an existing container, or from scratch. Building from scratch requires a recipe file (think of like a Dockerfile) and must be done with root permissions, which are not available on the ENCS infrastructure. Therefore, built-from-scratch containers must be created on a user-managed/personal system. There are three types of Singularity containers:

- File-system containers: built around the ext3 file system and are read-write "file", but cannot be resized once built.
- Sandbox containers: essentially a directory in an existing read-write space and are also read-write.
- Squashfs containers: read-only compressed "file" and are read-only. It is the default build type.

"A common workflow is to use the "sandbox" mode for container development and then build it as a default (squashfs) Singularity image when done." says the Singularity's authors about builds. File-system containers are considered legacy and are not commonly used.

For many workflows, a Docker container might already exist. In this case, you can use Singularity's docker pull function as part of your virtual environment setup in an interactive job allocation:

```
salloc --gpus=1 -n8 --mem=4Gb -t60
cd /speed-scratch/$USER/
singularity pull openiss-cuda-devicequery.sif docker://openiss/openiss-cuda-devicequery
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
```

This method can be used for converting Docker containers directly on Speed. On GPU nodes, make sure to pass on the `--nv` flag to Singularity so its containers could access the GPUs. See the linked example for more details.

3 Conclusion

The cluster operates on a "first-come, first-served" basis until it reaches full capacity. After that, job positions in the queue are determined based on past usage. The scheduler does attempt to fill gaps, so occasionally, a single-core job with lower priority may be scheduled before a multi-core job with higher priority.

3.1 Important Limitations

While Speed is a powerful tool, it is essential to recognize its limitations to use it effectively:

- New users are limited to a total of 32 cores and 4 GPUs. If you need more cores temporarily, please contact `rt-ex-hpc AT encs.concordia.ca`.
- Batch job sessions can run for a maximum of one week. Interactive jobs are limited to 24 hours see Section 2.8.
- Scripts can live in your NFS-provided home directory, but substantial data should be stored in your cluster-specific directory (located at `/speed-scratch/<USER>/`). NFS is suitable for short-term activities but not for long-term operations. Data that a job will read multiple times should be copied at the start to the scratch disk of a compute node using `$TMPDIR` (and possibly `$SLURM_SUBMIT_DIR`). Intermediate job data should be produced in `$TMPDIR`, and once a job is near completion, these data should be copied to your NFS-mounted home directory (or other NFS-mounted space). **In other words**,

IO-intensive operations should be performed locally whenever possible, reserving network activity for the start and end of jobs.

- Your current resource allocation is based on past usage, which considers approximately one week's worth of past wall clock time (time spent on the node(s)) and compute activity (on the node(s)).
- Jobs must always be run within the scheduler's system. Repeat offenders who run jobs outside the scheduler risk losing cluster access.

3.2 Tips/Tricks

- Ensure that files and scripts have Linux line breaks. Use the `file` command to verify and `dos2unix` to convert if necessary.
- Use `rsync` (preferred over `scp`) for copying or moving large amounts of data.
- Before transferring a large number of files between NFS-mounted storage and the cluster, compress the files into a `tar` archive.
- If you plan to use a different shell (e.g., `bash` [26]), change the shell declaration at the beginning of your script(s).
- Request resources (cores, memory, GPUs) that closely match the actual needs of your job. Requesting significantly more than necessary can make your job harder to schedule when resources are limited. Always check the efficiency of your job with either `seff` and/or the `--mail-type=ALL`, to adjust your job parameters.
- For any concerns or questions, email `rt-ex-hpc AT encs.concordia.ca`

3.3 Use Cases

- HPC Committee's initial batch about 6 students (end of 2019):
  - 10000 iterations job in Fluent finished in < 26 hours vs. 46 hours in Calcul Quebec
- NAG's MAC spoofer analyzer [20, 19], such as `https://github.com/smokhov/atsm/tree/master/examples/flucid`
  - compilation of forensic computing reasoning cases about false or true positives of hardware address spoofing in the labs
- S4 LAB/GIPSY R&D Group's:
  - MARFCAT and MARFPCAT (OSS signal processing and machine learning tools for vulnerable and weak code analysis and network packet capture analysis) [21, 17, 6]
  - Web service data conversion and analysis
  - Forensic Lucid encoders (translation of large log data into Forensic Lucid [18] for forensic analysis)
  - Genomic alignment exercises
- **Best Paper award**, Tariq Daradkeh, Gillian Roper, Carlos Alarcon Meza, and Serguei Mokhov. HPC jobs classification and resource prediction to minimize job failures. In

International Conference on Computer Systems and Technologies 2024 (CompSysTech '24), New York, NY, USA, June 2024. ACM

- Newton F. Ouedraogo and Ebenezer E. Essel. Unsteady wake interference of unequal-height tandem cylinders mounted in a turbulent boundary layer. *Journal of Fluid Mechanics*, 977:A52, 2023. https://doi.org/10.1017/jfm.2023.952
- Newton F. Ouedraogo and Ebenezer E. Essel. Effects of Reynolds number on the wake characteristics of a Notchback Ahmed body. *Journal of Fluids Engineering*, 146(11):111302, 05 2024
- L. Drummond, H. Banh, N. Ouedraogo, H. Ho, and E. Essel. Effects of nozzle convergence angle on the flow characteristics of a synthetic circular jet in a crossflow. In Bulletin of the American Physical Society, editor, *76th Annual Meeting of the Division of Fluid Dynamics*, November 2023
- N. Ouedraogo, A. Cyrus, and E. Essel. Effects of Reynolds number on the wake characteristics of a Notchback Ahmed body. In Bulletin of the American Physical Society, editor, *76th Annual Meeting of the Division of Fluid Dynamics*, November 2023
- Serguei Mokhov, Jonathan Llewellyn, Carlos Alarcon Meza, Tariq Daradkeh, and Gillian Roper. The use of containers in OpenGL, ML and HPC for teaching and research support. In *ACM SIGGRAPH 2023 Posters*, SIGGRAPH '23, New York, NY, USA, 2023. ACM. https://doi.org/10.1145/3588028.3603676
- Goutam Yelluru Gopal and Maria Amer. Separable self and mixed attention transformers for efficient object tracking. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, Hawaii, January 2024. https://arxiv.org/abs/2309.03979 and https://github.com/goutamyg/SMAT
- Goutam Yelluru Gopal and Maria Amer. Mobile vision transformer-based visual object tracking. In *34th British Machine Vision Conference (BMVC)*, Aberdeen, UK, November 2023. https://arxiv.org/abs/2309.05829 and https://github.com/goutamyg/MVT
- Farshad Rezaei and Marius Paraschivoiu. Computational challenges of simulating vertical axis wind turbine on the roof-top corner of a building. *Progress in Canadian Mechanical Engineering*, 6, 1–6 2023. http://hdl.handle.net/11143/20861
- Belkacem Belabes and Marius Paraschivoiu. CFD modeling of vertical-axis wind turbine wake interaction. *Transactions of the Canadian Society for Mechanical Engineering*, pages 1–10, 2023. https://doi.org/10.1139/tcsme-2022-0149
- Farshad Rezaei and Marius Paraschivoiu. Placing a small-scale vertical axis wind turbine on roof-top corner of a building. In *Proceedings of the CSME International Congress*, June 2022. https://doi.org/10.7939/r3-j7v7-m909
- Belkacem Belabes and Marius Paraschivoiu. CFD study of the aerodynamic performance of a vertical axis wind turbine in the wake of another turbine. In *Proceedings of the CSME International Congress*, 2022. https://doi.org/10.7939/r3-rker-1746
- Belkacem Belabes and Marius Paraschivoiu. Numerical study of the effect of turbulence intensity on VAWT performance. *Energy*, 233:121139, 2021. https://doi.org/10.1016/j.energy.2021.121139

- Parna Niksirat, Adriana Daca, and Krzysztof Skonieczny. The effects of reduced-gravity on planetary rover mobility. *International Journal of Robotics Research*, 39(7):797–811, 2020. `https://doi.org/10.1177/0278364920913945`
- The work "Haotao Lai. An OpenISS framework specialization for deep learning-based person re-identification. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, August 2019. `https://spectrum.library.concordia.ca/id/eprint/985788/`" using TensorFlow and Keras on OpenISS adjusted to run on Speed based on the repositories, and theirs forks by the team:
  - Haotao Lai et al. Openiss person re-identification baseline v0.1.1, June 2021. `https://github.com/OpenISS/openiss-reid-tfk` and
  - Haotao Lai et al. OpenISS keras-yolo3 v0.1.0, June 2021. `https://github.com/OpenISS/openiss-yolov3`

A History

A.1 Acknowledgments

- The first 6 to 6.5 versions of this manual and early UGE job script samples, Singularity testing,and user support were produced/done by Dr. Scott Bunnell during his time at Concordia as a part of the NAG/HPC group. We thank him for his contributions.
- The HTML version with devcontainer support was contributed by Anh H Nguyen.
- Dr. Tariq Daradkeh, was our IT Instructional Specialist from August 2022 to September 2023; working on the scheduler, scheduling research, end user support, and integration of examples, such as YOLOv3 in Section 2.15.4.2 and other tasks. We have a continued collaboration on HPC/scheduling research (see [8]).

A.2 Migration from UGE to SLURM

For long term users who started off with Grid Engine here are some resources to make a transition and mapping to the job submission process.

- Queues are called "partitions" in SLURM. Our mapping from the GE queues to SLURM partitions is as follows:

  ```
  GE  => SLURM
  s.q    ps
  g.q    pg
  a.q    pa
  ```

  We also have a new partition `pt` that covers SPEED2 nodes, which previously did not exist.

- Commands and command options mappings are found in Figure 13 from:
  `https://slurm.schedmd.com/rosetta.pdf`
  `https://slurm.schedmd.com/pdfs/summary.pdf`
  Other related helpful resources from similar organizations who either used SLURM for a while or also transitioned to it:
  `https://docs.alliancecan.ca/wiki/Running_jobs`

`https://www.depts.ttu.edu/hpcc/userguides/general_guides/Conversion_Table_1.pdf`
`https://docs.mpcdf.mpg.de/doc/computing/clusters/aux/migration-from-sge-to-slurm`

User Commands	PBS/Torque	Slurm	LSF	SGE
Job submission	qsub [script_file]	sbatch [script_file]	bsub [script_file]	qsub [script_file]
Job deletion	qdel [job_id]	scancel [job_id]	bkill [job_id]	qdel [job_id]
Job status (by job)	qstat [job_id]	squeue [job_id]	bjobs [job_id]	qstat -u * [-j job_id]
Job status (by user)	qstat -u [user_name]	squeue -u [user_name]	bjobs -u [user_name]	qstat [-u user_name]
Job hold	qhold [job_id]	scontrol hold [job_id]	bstop [job_id]	qhold [job_id]
Job release	qrls [job_id]	scontrol release [job_id]	bresume [job_id]	qrls [job_id]
Queue list	qstat -Q	squeue	bqueues	qconf -sql
Node list	pbsnodes -l	sinfo -N OR scontrol show nodes	bhosts	qhost
Cluster status	qstat -a	sinfo	bqueues	qhost -q
GUI	xpbsmon	sview	xlsf OR xlsbatch	qmon
Environment	**PBS/Torque**	**Slurm**	**LSF**	**SGE**
Job ID	$PBS_JOBID	$SLURM_JOBID	$LSB_JOBID	$JOB_ID
Submit Directory	$PBS_O_WORKDIR	$SLURM_SUBMIT_DIR	$LSB_SUBCWD	$SGE_O_WORKDIR
Submit Host	$PBS_O_HOST	$SLURM_SUBMIT_HOST	$LSB_SUB_HOST	$SGE_O_HOST
Node List	$PBS_NODEFILE	$SLURM_JOB_NODELIST	$LSB_HOSTS/LSB_MCPU_HOST	$PE_HOSTFILE
Job Array Index	$PBS_ARRAYID	$SLURM_ARRAY_TASK_ID	$LSB_JOBINDEX	$SGE_TASK_ID
Job Specification	**PBS/Torque**	**Slurm**	**LSF**	**SGE**
Script directive	#PBS	#SBATCH	#BSU	#$
Queue	-q [queue]	-p [queue]	-q	-q [queue]
Node Count	-l nodes=[count]	-N [min[-max]]	-	N/A
CPU Count	-l ppn=[count] OR -l mppwidth=[PE_count]	-n [count]	-n [count]	-pe [PE] [count]
Wall Clock Limit	-l walltime=[hh:mm:ss]	-t [min] OR -t [days-hh:mm:ss]	-W [hh:mm:ss]	-l h_rt=[seconds]
Standard Output FIle	-o [file_name]	-o [file_name]	-o [file_name]	-o [file_name]
Standard Error File	-e [file_name]	e [file_name]	-e [file_name]	-e [file_name]
Combine stdout/err	-j oe (both to stdout) OR -j eo (both to stderr)	(use -o without -e)	(use -o without -e)	-j yes
Copy Environment	-V	--export=[ALL | NONE | variables]		-V
Event Notification	-m abe	--mail-type=[events]	-B or -N	-m abe
Email Address	-M [address]	--mail-user=[address]	-u [address]	-M [address]
Job Name	-N [name]	--job-name=[name]	-J [name]	-N [name]
Job Restart	-r [y|n]	--requeue OR --no-requeue (NOTE: configurable default)	-r	-r [yes|no]
Working Directory	N/A	--workdir=[dir_name]	(submission directory)	-wd [directory]
Resource Sharing	-l naccesspolicy=singlejob	--exclusive OR--shared	-x	-l exclusive
Memory Size	-l mem=[MB]	--mem=[mem][M|G|T] OR --mem-per-cpu=[mem][M|G|T]	-M [MB]	-l mem_free=[memory][K|M|G]
Account to charge	-W group_list=[account]	--account=[account]	-P [account]	-A [account]
Tasks Per Node	-l mppnppn [PEs_per_node]	--tasks-per-node=[count]		(Fixed allocation_rule in PE)
CPUs Per Task		--cpus-per-task=[count]		
Job Dependency	-d [job_id]	--depend=[state:job_id]	-w [done | exit | finish]	-hold_jid [job_id | job_name]
Job Project		--wckey=[name]	-P [name]	-P [name]
Job host preference		--nodelist=[nodes] AND/OR --exclude=[nodes]	-m [nodes]	-q [queue]@[node] OR -q [queue]@@[hostgroup]
Quality Of Service	-l qos=[name]	--qos=[name]		
Job Arrays	-t [array_spec]	--array=[array_spec] (Slurm version 2.6+)	J "name[array_spec]"	-t [array_spec]
Generic Resources	-l other=[resource_spec]	--gres=[resource_spec]		-l [resource]=[value]
Licenses		--licenses=[license_spec]	-R "rusage[license_spec]"	-l [license]=[count]
Begin Time	-A "YYYY-MM-DD HH:MM:SS"	--begin=YYYY-MM-DD[THH:MM[:SS]]	-b[[year:][month:]daty:]hour:minute	-a [YYMMDDhhmm]

Figure 13: Rosetta Mappings of Scheduler Commands from SchedMD

- **NOTE:** If you have used UGE commands in the past you probably still have these lines there; **they should now be removed**, as they have no use in SLURM and will start giving "command not found" errors on login when the software is removed:

  csh/`tcsh`: sample `.tcshrc` file:

  ```
  # Speed environment set up
  if ($HOSTNAME == speed-submit.encs.concordia.ca) then
  source /local/pkg/uge-8.6.3/root/default/common/settings.csh
  endif
  ```

  Bourne shell/`bash`: sample `.bashrc` file:

```
# Speed environment set up
if [ $HOSTNAME = "speed-submit.encs.concordia.ca" ]; then
    . /local/pkg/uge-8.6.3/root/default/common/settings.sh
    printenv ORGANIZATION | grep -qw ENCS || . /encs/Share/bash/profile
fi
```

IMPORTANT NOTE: you will need to either log out and back in, or execute a new shell, for the environment changes in the updated `.tcshrc` or `.bashrc` file to be applied.

A.3 Phases

Brief summary of Speed evolution phases:

A.3.1 Phase 5

Phase 5 saw incorporation of the Salus, Magic, and Nebular subclusters (see Figure 2).

A.3.2 Phase 4

Phase 4 had 7 SuperMicro servers with 4x A100 80GB GPUs each added, dubbed as "SPEED2". We also moved from Grid Engine to SLURM.

A.3.3 Phase 3

Phase 3 had 4 vidpro nodes added from Dr. Amer totalling 6x P6 and 6x V100 GPUs added.

A.3.4 Phase 2

Phase 2 saw 6x NVIDIA Tesla P6 added and 8x more compute nodes. The P6s replaced 4x of FirePro S7150.

A.3.5 Phase 1

Phase 1 of Speed was of the following configuration:

- Sixteen, 32-core nodes, each with 512 GB of memory and approximately 1 TB of volatile-scratch disk space.
- Five AMD FirePro S7150 GPUs, with 8 GB of memory (compatible with the Direct X, OpenGL, OpenCL, and Vulkan APIs).

B Frequently Asked Questions

B.1 Where do I learn about Linux?

All Speed users are expected to have a basic understanding of Linux and its commonly used commands. Here are some recommended resources:

Software Carpentry : Software Carpentry provides free resources to learn software, including a workshop on the Unix shell. Visit Software Carpentry Lessons to learn more.

Udemy : There are numerous Udemy courses, including free ones, that will help you learn Linux. Active Concordia faculty, staff and students have access to Udemy courses. A recommended starting point for beginners is the course "Linux Mastery: Master the Linux Command Line in 11.5 Hours". Visit Concordia's Udemy page to learn how Concordians can access Udemy.

B.2 How to use bash shell on Speed?

This section provides comprehensive instructions on how to utilize the bash shell on the Speed cluster.

B.2.1 How do I set bash as my login shell?

To set your default login shell to bash on Speed, your login shell on all GCS servers must be changed to bash. To make this change, create a ticket with the Service Desk (or email `help at concordia.ca`) to request that bash become your default login shell for your ENCS user account on all GCS servers.

B.2.2 How do I move into a bash shell on Speed?

To move to the bash shell, type **bash** at the command prompt:

```
[speed-submit] [/home/a/a_user] > bash
bash-4.4$ echo $0
bash
```

Note how the command prompt changes from "`[speed-submit] [/home/a/a_user] >`" to "`bash-4.4$`" after entering the bash shell.

B.2.3 How do I use the bash shell in an interactive session on Speed?

Below are examples of how to use `bash` as a shell in your interactive job sessions with both the `salloc` and `srun` commands.

- `salloc -ppt --mem=100G -N 1 -n 10 /encs/bin/bash`
- `srun --mem=50G -n 5 --pty /encs/bin/bash`

Note: Make sure the interactive job requests memory, cores, etc.

B.2.4 How do I run scripts written in bash on `Speed`?

To execute bash scripts on Speed:

1. Ensure that the shebang of your bash job script is `#!/encs/bin/bash`
2. Use the `sbatch` command to submit your job script to the scheduler.

Check Speed GitHub for a sample bash job script.

B.3 How to resolve "Disk quota exceeded" errors?

B.3.1 Probable Cause

The "`Disk quota exceeded`" error occurs when your application has run out of disk space to write to. On `Speed`, this error can be returned when:

1. The NFS-provided home is full and cannot be written to. You can verify this using the `quota` and `bigfiles` commands.
2. The "`/tmp`" directory on the speed node where your application is running is full and cannot be written to.

B.3.2 Possible Solutions

1. Use the `--chdir` job script option to set the job working directory. This is the directory where the job will write output files.
2. Although local disk space is recommended for IO-intensive operations, the '`/tmp`' directory on `Speed` nodes is limited to 1TB, so it may be necessary to store temporary data elsewhere. Review the documentation for each module used in your script to determine how to set working directories. The basic steps are:
   - Determine how to set working directories for each module used in your job script.
   - Create a working directory in `speed-scratch` for output files:

     ```
     mkdir -m 750 /speed-scratch/$USER/output
     ```

   - Create a subdirectory for recovery files:

     ```
     mkdir -m 750 /speed-scratch/$USER/recovery
     ```

   - Update the job script to write output to the directories created in your `speed-scratch` directory, e.g., `/speed-scratch/$USER/output`.

In the above example, `$USER` is an environment variable containing your ENCS username.

B.3.3 Example of setting working directories for `COMSOL`

- Create directories for recovery, temporary, and configuration files.

  ```
  mkdir -m 750 -p /speed-scratch/$USER/comsol/{recovery,tmp,config}
  ```

- Add the following command switches to the COMSOL command to use the directories created above:

  ```
  -recoverydir /speed-scratch/$USER/comsol/recovery
  -tmpdir /speed-scratch/$USER/comsol/tmp
  -configuration/speed-scratch/$USER/comsol/config
  ```

In the above example, `$USER` is an environment variable containing your ENCS username.

B.3.4 Example of setting working directories for `Python Modules`

By default when adding a Python module, the `/tmp` directory is set as the temporary repository for files downloads. The size of the `/tmp` directory on `speed-submit` is too small for PyTorch. To add a Python module

- Create your own tmp directory in your `speed-scratch` directory:

  ```
  mkdir /speed-scratch/$USER/tmp
  ```

- Use the temporary directory you created

```
setenv TMPDIR /speed-scratch/$USER/tmp
```

- Attempt the installation of PyTorch

In the above example, `$USER` is an environment variable containing your ENCS username.

B.4 How do I check my job's status?

When a job with a job ID of 1234 is running or terminated, you can track its status using the following commands to check its status:

- Use the "sacct" command to view the status of a job:

  ```
  sacct -j 1234
  ```

- Use the "squeue" command to see if the job is sitting in the queue:

  ```
  squeue -j 1234
  ```

- Use the "sstat" command to find long-term statistics on the job after it has terminated and the `slurmctld` has purged it from its tracking state into the database:

  ```
  sstat -j 1234
  ```

B.5 Why is my job pending when nodes are empty?

B.5.1 Disabled nodes

It is possible that one or more of the Speed nodes are disabled for maintenance. To verify if Speed nodes are disabled, check if they are in a draining or drained state:

```
[serguei@speed-submit src] % sinfo --long --Node
Thu Oct 19 21:25:12 2023
NODELIST   NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE REASON
speed-01       1        pa        idle 32     2:16:1 257458        0      1    gpu16 none
speed-03       1        pa        idle 32     2:16:1 257458        0      1    gpu32 none
speed-05       1        pg        idle 32     2:16:1 515490        0      1    gpu16 none
speed-07       1       ps*       mixed 32     2:16:1 515490        0      1    cpu32 none
speed-08       1       ps*    drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-09       1       ps*    drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-10       1       ps*    drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-11       1       ps*       idle 32     2:16:1 515490        0      1    cpu32 none
speed-12       1       ps*    drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-15       1       ps*    drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-16       1       ps*    drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-17       1        pg    drained 32     2:16:1 515490        0      1    gpu16 UGE
speed-19       1       ps*       idle 32     2:16:1 515490        0      1    cpu32 none
speed-20       1       ps*    drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-21       1       ps*    drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-22       1       ps*    drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-23       1       ps*       idle 32     2:16:1 515490        0      1    cpu32 none
speed-24       1       ps*       idle 32     2:16:1 515490        0      1    cpu32 none
speed-25       1        pg       idle 32     2:16:1 257458        0      1    gpu32 none
speed-25       1        pa       idle 32     2:16:1 257458        0      1    gpu32 none
speed-27       1        pg       idle 32     2:16:1 257458        0      1    gpu32 none
speed-27       1        pa       idle 32     2:16:1 257458        0      1    gpu32 none
```

```
speed-29       1       ps*        idle 32     2:16:1 515490        0      1    cpu32 none
speed-30       1       ps*     drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-31       1       ps*     drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-32       1       ps*     drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-33       1       ps*        idle 32     2:16:1 515490        0      1    cpu32 none
speed-34       1       ps*        idle 32     2:16:1 515490        0      1    cpu32 none
speed-35       1       ps*     drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-36       1       ps*     drained 32     2:16:1 515490        0      1    cpu32 UGE
speed-37       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-38       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-39       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-40       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-41       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-42       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
speed-43       1        pt        idle 256    2:64:2 980275        0      1 gpu20,mi none
```

Note which nodes are in the state of **drained**. The reason for the drained state can be found in the **reason** column. Your job will run once an occupied node becomes availble or the maintenance is completed, and the disabled nodes have a state of **idle**.

B.5.2 Error in job submit request.

It is possible that your job is pending because it requested resources that are not available within Speed. To verify why job ID 1234 is not running, execute:

```
sacct -j 1234
```

A summary of the reasons can be obtained via the `squeue` command.

C Sister Facilities

Below is a list of resources and facilities similar to Speed at various capacities. Depending on your research group and needs, they might be available to you. They are not managed by HPC/NAG of AITS, so contact their respective representatives.

- `computation.encs` is a CPU-only 3-machine cluster running longer jobs without a scheduler at the moment. Shares the same EL7 software tree as Speed's EL7 nodes as well as lab desktops. See `https://www.concordia.ca/ginacody/aits/public-servers.html`.
- `apini.encs` cluster for teaching and MPI programming (see the corresponding course in CSSE), managed by CSSE.
- Computer Science and Software Engineering (CSSE) Virya GPU Cluster. For CSSE members only. The cluster has 4 nodes with total of 32 NVIDIA GPUs (a mix of V100s and A100s). To request access send email to `virya.help AT concordia.ca`. This includes an Atlas Analytics partition of Dr. Mahdi Husseini.
- Dr. Eugene Belilovsky hightower Exxact, and megatower graphcore clusters.
- Dr. Maria Amer's VidPro group's nodes in Speed (-01, -03, -25, -27) with additional V100 and P6 GPUs.

- There are various Lambda Labs other GPU servers and like computers acquired by individual researchers; if you are member of their research group, contact them directly. These resources are not managed by us.
  - Dr. Amin Hammad's `construction.encs` Lambda Labs station
  - Dr. Hassan Rivaz's `impactlab.encs` Lambda Labs station
  - Dr. Nizar Bouguila's `xailab.encs` Lambda Labs station
  - Dr. Roch Glitho's `femto.encs` server
  - Dr. Maria Amer's `venom.encs` Lambda Labs station
  - Dr. Leon Wang's `guerrera.encs` DGX station
- Dr. Ivan Contreras' 4 Operations Research group servers (managed by AITS).
- If you are a member of School of Health (formerly PERFORM Center), you may have access to their local PERFORM's High Performance Computing (HPC) Cluster. Contact Thomas Beaudry for details and how to obtain access.
- All Concordia students have access to the Library's small Technology Sandbox testing cluster that also runs Slurm. Email `sean.cooney AT concordia.ca` for details.
- Digital Research Alliance Canada (Compute Canada / Calcul Quebec), `https://alliancecan.ca/`. Follow this link on the information how to obtain access (students need to be sponsored by their supervising faculty members, who should create accounts first). Their SLURM examples are here: `https://docs.alliancecan.ca/wiki/Running_jobs`

D Software Installed On Speed

This is a generated section by a script; last updated on *Fri Dec 20 04:37:31 PM EST 2024*. We have two major software trees: Scientific Linux 7 (EL7), which is outgoing, and AlmaLinux 9 (EL9). After major synchronization of software packages is complete, we will stop maintaining the EL7 tree and will migrate the remaining nodes to EL9.

Use `--constraint=el7` to select EL7-only installed nodes for their software packages. Conversely, use `--constraint=el9` for the EL9-only software. These options would be used as a part of your job parameters in either `#SBATCH` or on the command line.

NOTE: this list does not include packages installed directly on the OS (yet).

D.1 EL7

Not all packages are intended for HPC, but the common tree is available on Speed as well as teaching labs' desktops.

- `a2ps-4.13b`
- `a2ps-4.14`
- `abaqus-2019`
- `abaqus-2020`
- `abaqus-2021`
- `abaqus-2023`
- `acl-10.0.express`
- `acl-10.1.express`
- `acroread-9.5.5`
- `ADS-2016.01`
- `ADS-2017.01`
- `ADS-2019`

- ADS-2020u1
- adt_bundle-20140702
- alpine-2.00
- alpine-2.25
- anaconda-1.7.0
- anaconda2-2019.07
- anaconda2-5.1.0
- anaconda3-2019.07
- anaconda3-2019.10
- anaconda3-2021.05
- anaconda3-2023.03
- anaconda3-5.1.0
- android_sdk-24.4.1
- android_studio-162.4069837
- android_studio-173.4720617
- ansoft_designer-6.1.2
- ansoft_designer-8.0
- ansys-16.2
- ansys-17.2
- ansys-18.2
- ansys-19.2
- ansys-2019R3
- ansys-2020R2
- ansys-2021R1
- ansys-2021R2
- ansys-2022R1
- ansys-2022R2
- ansys-2023R1
- ansys-2023R2
- AnsysEM-16.2
- ant-1.10.11
- ant-1.10.2
- ant-1.6.2
- ant-1.9.7
- ANTs-2.3.5
- ApacheDirectoryStudio-1.5.3
- arduino-1.6.8
- ArgoUML-0.34
- arpack-3.0.2
- ARWpost-3.1
- aspectj-1.7.0.M1
- aspectj-1.8.2
- aspectj-1.8.6
- aspell-0.60.6
- aspell-0.60.8
- atanua-1.2.130617
- autoconf-2.68
- autoconf-2.71
- AutoDock-4.2.6
- autogen-5.18.4
- automake-1.13.1
- automake-1.16.1
- autoson-1.4.3
- babl-0.1.12
- bash-4.4
- basilisk-19_3_23
- bazel-0.2.0
- bazel-0.21.0
- bazel-0.24.1
- bazel-0.26.1
- bazel-2.0.0
- bazel-3.7.2
- bazel-4.2.1
- bison-2.7.1
- bison-3.7.2
- blas-3.10.0
- blender-2.66a
- blender-2.78
- blender-2.78c
- bogofilter-1.2.4
- booksim-2.0
- boost-1.51.0
- boost-1.62.0
- boost-1.68.0
- boost-1.69.0
- boost-1.70.0
- boost-1.73.0
- bouncer-2.2
- buddy-2.4
- bzr-2.6.0
- camlp5-6.02.3
- camlp5-6.14
- CGAL-4.3
- cgiwrap-4.1
- Check-0.15.2
- chrome-101.0.4951.54
- chromium-31.0.1650.63
- cilkplus-4.8
- clips-6.23
- clips-6.30
- clojure-1.8.0
- cmake-2.8.12
- cmake-3.18.4
- cmake-3.26.4
- cmake-3.8.2
- CMC_utils-1.0
- codeblocks-13.12
- codelite-9.1
- comsol-3.5a
- comsol-4.0
- comsol-4.1
- comsol-5.6
- comsol-6.0
- comsol-6.1
- concorde-20031219
- coware-2010.1
- cplex-12.10.0
- cplex-12.6.1
- cplex-12.6.2
- cplex-12.6.3
- cplex-12.7.0
- cplex-12.7.1
- cplex-12.8.0
- cplex-12.9.0
- cplex-20.1.0
- cplex-22.1.0
- cplex-22.1.1
- cppunit-1.13.2
- CST-2014
- CST-2019
- CST-2020
- cuda-10.0
- cuda-10.2

- cuda-11.5
- cuda-11.8
- cuda-9.2
- cups-1.3.7
- cups-2.3.3
- curl-7.15.0
- curl-7.38.0
- curl-7.73.0
- curl-7.84.0
- curl-7.86.0
- curl-7.87.0
- curl-8.1.2
- cvs-1.11.23
- cvx-2.1
- DbVisualizer-24.1.5
- DbVisualizer-9.1.9
- ddd-3.3.12
- dejagnu-1.5.3
- dejagnu-1.6.2
- digilent-2.19.2
- digilent-2.8.2
- dmtcp-1.2.8
- dos2unix-7.3
- doxygen-1.8.4
- drush-6.5.0
- ece-eclipse-jee.juno
- ece-eclipse-jee.luna
- ecj-25
- ECLiPSe-6.0_192
- eclipse-jee.201812
- eclipse-jee.202003
- eclipse-jee.202109
- eclipse-jee.202209
- eclipse-jee.indigo
- eclipse-jee.juno
- eclipse-jee.luna
- eclipse-jee.mars
- eclipse-jee.neon
- eclipse-jee.oxygen
- eigen-3.3.7
- electromagnetics_suite-2022R2
- electromagnetics_suite-2023R1
- emacs-24.4
- emacs-25.2
- enscript-1.6.5.2
- exmh-2.7.0
- expat-2.1.0
- expect-5.45
- expect-5.45.4
- fanout-0.6.1
- feko-2017.2.2
- feko-2018.2
- feko-6.1
- feko-7.0.1
- ffmpeg-0.10.2
- ffmpeg-3.3.2
- ffmpeg-4.1.3
- fftw-3.2.2
- fftw-3.3.10
- fftw-3.3.8
- firefox-102.11.0
- firefox-102.12.0
- firefox-102.13.0
- firefox-102.14.0
- firefox-102.15.0
- firefox-102.15.1
- firefox-115.10.0
- firefox-115.2.1
- firefox-115.3.0
- firefox-17.0.11
- firefox-2.0.0.20
- firefox-24.8.1
- firefox-31.7.0
- firefox-38.8.0
- firefox-45.9.0
- firefox-52.9.0
- firefox-60.9.0
- firefox-68.12.0
- firefox-78.15.0
- firefox-91.11.0
- firefox_french-102.11.0
- firefox_french-102.12.0
- firefox_french-102.13.0
- firefox_french-102.14.0
- firefox_french-102.15.0
- firefox_french-102.15.1
- firefox_french-115.10.0
- firefox_french-115.2.1
- firefox_french-115.3.0
- firefox_french-24.8.1
- firefox_french-31.7.0
- firefox_french-38.8.0
- firefox_french-45.9.0
- firefox_french-52.9.0
- firefox_french-60.9.0
- firefox_french-68.12.0
- firefox_french-78.15.0
- firefox_french-91.11.0
- flex-2.5.37
- flex-2.6.4
- fltk-1.3.2
- fltk-1.3.8
- fox-1.6.49
- FreeImage-3.18.0
- freerdp-1.0.2
- freerdp-1.2.0
- freetype-2.4.12
- FSL-6.0.5
- fsl-6.0.6.2
- fsr-1.9
- fxscintilla-2.28.0
- gambit-c-4.7.5
- gate-7.1
- gc-7.2f
- gcc-12.2.0
- gcc-13.1.0
- gcc-3.3.2
- gcc-4.1.2
- gcc-4.4.3
- gcc-4.7.2
- gcc-4.9.2
- gcc-5.1.0
- gcc-5.2.0

- gcc-5.4.0
- gcc-6.1.0
- gcc-7.3.0
- gcc-8.4.0
- gcc-9.2.0
- gcc-9.3.0
- gcc-arm-11.2.2022.02
- gdb-12.1
- gdb-7.7
- geomview-1.9.4
- gerris-131206
- getmail-4.54.0
- gettext-0.18.1.1
- gfsview-121130
- ghc-7.6.3
- ghostscript-10.0.0
- ghostscript-8.50
- ghostscript-9.25
- ghostscript-9.26
- ghostscript-9.50
- ghostscript-9.52
- gifsicle-1.92
- gimp-2.8.14
- gimp-2.8.18
- git-1.8.5.3
- git-2.15.0
- git-2.22.0
- git-2.23.0
- git-2.23.3
- git-2.29.2
- git-2.35.1
- git-2.5.0
- gl2ps-1.3.8
- glade3-3.8.3
- glew-1.10.0
- glew-2.1.0
- glfw-3.0.4
- glfw-3.3
- glfw-3.3.4
- glm-0.9.5.4
- glm-0.9.9.8
- glpk-4.55
- gmp-4.3.2
- gnome_env-20110618
- gnome_env-20130510
- gnupg-2.3.4
- gnuplot-4.6.3
- gnuplot-5.0.1
- gnuplot-5.0.3
- go-1.10.2
- go-1.12
- go-1.15.6
- go-1.19.3
- go-1.3
- go-1.9
- GoldenGate-2020u1
- gperf-3.0.4
- grace-5.1.25
- GraphiteTwo-a5
- GraphiteTwo-Concordia_v1
- graphviz-2.30.0
- graphviz-2.40.1
- gromacs-5.0.7
- gsl-2.6
- gtkwave-3.3.15
- gts-121130
- guile-2.0.11
- gurobi-10.0.0
- gurobi-10.0.1
- gurobi-7.0.1
- gurobi-7.5.0
- gurobi-8.0.0
- gurobi-8.1.0
- gurobi-9.0.0
- gurobi-9.0.2
- gurobi-9.1.0
- gurobi-9.5.0
- gurobi-9.5.2
- gv-3.7.4
- hadoop-1.0.4
- hadoop-1.1.2
- hadoop-2.4.1
- hadoop-2.7.1
- hadoop-2.7.3
- hadoop-2.9.0
- haskell_platform-2013.2.0.0
- hdf5-1.10.5
- hdf5-1.8.11
- hexpert-2.4.1
- hmmer-3.3.2
- html2ps-1.0b5
- httpd-2.2.22
- httpd-2.2.32
- httpd-2.4.27
- httpd-2.4.38
- httpd-2.4.39
- httpd-2.4.52
- httpd-2.4.53
- httpd-2.4.54
- httpd-2.4.55
- httpd-2.4.57
- httpd-current
- http-parser-2.9.4
- hunspell-1.7.2
- hwloc-1.11.2
- hwloc-2.8.0
- hxplay-11.0.2
- hyperworks-12.0
- iBioSim-3.0.0.beta
- ImageMagick-6.3.6
- ImageMagick-6.9.4.1
- ImageMagick-7.0.8.42
- imap-2007e
- instantclient-19.3.0.0.0
- IntelliJ-2019.2.3
- intltool-0.50.2
- invtools-2.0
- invtools-2.0.2
- invtools-2.1.0
- invtools-2.2.0
- invtools-2.3.0
- invtools-2.4.0
- invtools-2.5.0

- invtools-2.5.1
- invtools-2.6.0
- invtools-2.7.0
- invtools-2.8.0
- invtools-3.0.0
- invtools-3.1.0
- ipe-7.1.3
- iq-8.4.1
- j2sdk_32b-1.5.0_22
- j2sdk_64b-1.5.0_22
- jansson-2.14
- jdk-11
- jdk-11.0.1
- jdk-11.0.2
- jdk-17
- jdk-17.0.2
- jdk-19
- jdk-19.0.2
- jdk_32b-6u45
- jdk_32b-7u80
- jdk_32b-8u101
- jdk_32b-8u111
- jdk_32b-8u121
- jdk_32b-8u131
- jdk_32b-8u141
- jdk_32b-8u151
- jdk_32b-8u161
- jdk_32b-8u171
- jdk_32b-8u181
- jdk_32b-8u191
- jdk_32b-8u201
- jdk_32b-8u211
- jdk_32b-8u231
- jdk_32b-8u91
- jdk-5
- jdk-6
- jdk-6_32b
- jdk_64b-6u45
- jdk_64b-7u80
- jdk_64b-8u101
- jdk_64b-8u111
- jdk_64b-8u121
- jdk_64b-8u131
- jdk_64b-8u141
- jdk_64b-8u151
- jdk_64b-8u161
- jdk_64b-8u171
- jdk_64b-8u181
- jdk_64b-8u191
- jdk_64b-8u201
- jdk_64b-8u211
- jdk_64b-8u231
- jdk_64b-8u91
- jdk-7
- jdk-7_32b
- jdk-8
- jdk-8_32b
- jes-5.02
- jmag-20.1.02zi
- json-c-0.16
- julia-1.7.2
- kaldi-5.5
- kerberos-helpers-1.0
- kicad-4.0.1
- kile-2.0.1
- lam-7.1.3
- lapack-3.10.0
- lapack-3.4.2
- lasso-2.6.0
- lib3ds-1.3.0
- libarchive-3.1.2
- libarchive-3.3.2
- libassuan-2.5.5
- libatomic_ops-7.4.0
- libav-12.3
- libedit-20210910.3.1
- libevent-2.1.12
- libffi-3.2.1
- libgcrypt-1.10.1
- libgd-2.2.5
- libglade-2.6.4
- libgpg-error-1.44
- libidn2-2.3.2
- libjwt-1.15.2
- libksba-1.6.0
- LibreOffice-7.1.8
- LibreOffice-7.3.7
- LibreOffice-7.4.7
- libssh-0.7.3
- libssh-0.7.6
- libtool-2.4.6
- libunistring-0.9.6
- libusb-1.0.8
- libxml2-2.7.6
- libxml2-2.7.8
- libxml2-2.9.4
- libyaml-0.2.5
- libzip-1.5.1
- lispworks-6.1
- lispworks-7.1
- lispworks-8.0
- Logiscope-6.6.0
- Logiscope-7.1
- lshw-B.02.16
- ltct-20060701
- lua-5.1.5
- lua-5.3.4
- lucene-4.4.0
- lucene-6.6.0
- LWkit-2.0
- lynx-2.8.7
- lynx-2.8.9
- lynx-2.8.9dev.11
- lyx-2.0.3
- lz4-1.9.4
- m4-1.4.17
- mairix-0.23
- make-4.4.1
- maple-2019.1
- maple-2022.1
- matlab-R0223b
- matlab-R2020b
- matlab-R2021b

- matlab-R2022a
- matlab-R2022b
- matlab-R2023a
- matlab-R2023b
- maven-3.3.9
- maven-3.6.3
- mcmas-1.0.1
- mcmas-1.3.0
- mercurial-3.0.2
- mercurial-5.6.1
- mesa-19.0.3
- metamail-2.7
- metis-5.1.0
- mininet-2.3.0
- MKL-2021.04
- mod_auth_kerb-5.4
- mod_auth_mellon-0.17.0
- mod_authnz_external-3.2.5
- mod_authnz_external-3.3.2
- mod_authz_unixgroup-1.0.2
- mod_authz_unixgroup-1.1.0
- modeFRONTIER-2017R2
- mod_perl-2.0.10
- mod_perl-2.0.5
- modules-3.2.10
- modules-4.6.1
- modules-5.3.1
- modules-current
- mongodb-2.6.5
- mongodb-3.2.10
- mongodb-3.4.6
- mono-3.0.11
- mono-4.4.2.8
- mono-6.12.0.122
- monodevelop-3.1.1
- monodevelop-5.10.0
- monodevelop-7.8.4
- mosek-7.1.0.54
- mosek-8.1.0.58
- mpack-1.6
- mpc-1.0.1
- mpfr-2.4.2
- mpich-4.1
- MPlayer-1.1.1
- mplayerplugin-3.45
- mpm-itk-2.4.7
- MRtrix-3.0.3
- MUMPS-5.5.0
- mysql-5.1.66
- mysql-5.6.17
- mysql-5.6.37
- mysql-5.6.39
- mysql-5.6.42
- mysql-5.6.43
- mysql-5.7.20
- mysql-5.7.36
- mysql-8.0.16
- mysql-8.0.18
- mysql-8.0.22
- mysql-8.0.23
- mysql-8.0.31
- mysql_workbench-6.3.3
- mysql_workbench-8.0.16
- nagtools-2.1.10
- nagtools-2.1.3
- nagtools-2.1.4
- nagtools-2.1.5
- nagtools-2.1.6
- nagtools-2.1.7
- nagtools-2.1.8
- nagtools-2.1.9
- nano-6.2
- nasm-2.10.07
- nasm-2.14.02
- nasm-2.15.05
- ncl-6.6.2
- ncurses-6.4
- netbeans-7.3.1
- netbeans-8.0.2
- netbeans-8.1
- netcdf-4.1.3
- netcdf-4.3.0
- netcdf_c-4.7.2
- netcdf_fortran-4.5.2
- netpbm-10.35.95
- net-snmp-5.4.1
- net-snmp-5.9.1
- nettle-2.7.1
- ninja-1.10.2
- nltk-3.0
- nltk-3.3
- nmh-1.1rc3
- nmh-1.5
- nmh-1.6
- nmh-1.7.1
- node-v0.10.33
- node-v12.13.0
- node-v12.16.1
- node-v12.18.0
- node-v16.13.0
- node-v7.8.0
- node-v9.5.0
- nph-1.2.2
- npth-1.6
- ns-3.26
- ntbtls-0.3.0
- nvtop-3.0.1
- ocaml-3.12.1
- ocaml-4.01.0
- octave-3.8.2
- octave-4.0.3
- octave-7.3.0
- octave-8.2.0
- omnetpp-4.2.2
- omnetpp-5.6.2
- oniguruma-6.9.5
- Open3D-0.11.1
- opencv-3.0.0
- opencv-3.3.0
- opencv-3.4.5
- opencv-4.0.1
- opencv-4.5.0
- opencv-4.5.4

- openesb-2.3.1
- openesb-3.0.5
- OpenFOAM-10.0
- OpenFOAM-11.0
- OpenFOAM-1.7.1
- OpenFOAM-2.3.1
- OpenFOAM-2.4.0
- OpenFOAM-3.0.1
- OpenFOAM-5.0
- OpenFOAM-6.0
- OpenFOAM-7.0
- OpenFOAM-8.0
- OpenFOAM-9.0
- OpenFOAM-v2012
- OpenFOAM-v2106
- OpenFOAM-v2206
- OpenFOAM-v2212
- OpenFOAM-v2306
- openjpeg-2.3.1
- openmotif-2.2.4
- openmpi-1.6.3
- openmpi-1.8.3
- openmpi-3.1.3
- openmpi-4.0.1
- openocd-0.11.0
- openpmix-5.0.1
- OpenSees-3.4.0
- openssl-1.0.2.current
- openssl-1.0.2l
- openssl-1.0.2o
- openssl-1.0.2u
- openssl-1.1.1a
- openssl-1.1.1.current
- openssl-1.1.1g
- openssl-1.1.1j
- openssl-1.1.1n
- openssl-1.1.1o
- openssl-1.1.1p
- openssl-1.1.1q
- openssl-1.1.1s
- openssl-1.1.1t
- openssl-1.1.1u
- openssl-3.0.10
- openssl-3.0.11
- openssl-3.0.12
- openssl-3.0.4
- openssl-3.0.5
- openssl-3.0.6
- openssl-3.0.7
- openssl-3.0.8
- openssl-3.0.9
- openssl-3.0.current
- openssl-3.1.0
- openssl-3.1.current
- opera-12.16
- oracle-19c
- os-overrides-1.0
- otp-20.1
- p11_kit-0.22.1
- p7zip-16.02
- p7zip-9.20.1
- parallel-20200322
- ParaView-5.11.2
- parmetis-4.0.3
- pc2-9.3.1
- pcre-8.21
- pdfedit-0.4.5
- perl-5.28.1
- perl-5.28.1_mt
- perl-5.30.3
- perl-5.30.3_mt
- perl-5.8.3
- perl-5.8.3_mt
- pgadmin3-1.10.2
- pgadmin4-5.0
- pgadmin4-7.6
- php-5.3.15
- php-5.5.11
- php-5.5.12
- php-5.5.38
- php-5.6.33
- php-5.6.40
- php-7.2.11
- php-7.2.4
- php-7.2.5
- php-7.4.11
- php-7.4.14
- php-7.4.27
- php-7.4.33
- php-8.2.6
- ph-substitute-1.0
- pinentry-1.2.0
- poppler-0.81.0
- postgresql-11.7
- postgresql-12
- postgresql-12.2
- postgresql-12.3
- postgresql-8.0.15
- postgresql-8.3.18
- postgresql-9.6.8
- praat-6.3.06
- Precision-2010aU1
- print-utils-1.0
- probreg-0.3.1
- proj-6.0.0
- protege-4.0
- protobuf-2.5.0
- pwauth-2.3.11
- pwauth-2.3.8
- python-2.5.6
- python-2.7.10
- python-2.7.11
- python-3.10.13
- python-3.10.6
- python-3.11.0
- python-3.11.5
- python-3.11.6
- python-3.12.0
- python-3.2.3
- python-3.3.0
- python-3.4.3
- python-3.5.1
- python-3.6.15

- python-3.7.3
- python-3.7.7
- python-3.8.18
- python-3.8.3
- python-3.8.9
- python-3.9.1
- python-3.9.18
- pytorch-1.1.0
- pytorch-1.10.0
- pytorch-1.6.0
- qemu-6.1.1
- qhull-2012.1
- qhull-2020.2
- qrupdate-1.1.2
- QScintilla-2.8.4
- qt-4.8.6
- qt-5.14.2
- qt-5.9.6
- qt_sdk-1.2.1
- quota-1.3
- R-3.4.1
- R-3.4.2
- R-3.5.0
- R-3.6.1
- R-4.1.3
- R-4.2.2
- racket-8.3
- rapidjson-1.1.0
- rational_rose_realtime-7.0
- rdesktop-1.8.3
- rdesktop-1.8.4
- readline-5.2
- redis-4.0.10
- redis-4.0.8
- RSA-8.5
- ruby-2.1.1
- ruby-2.3.1
- ruby-2.5.0
- ruby-2.5.3
- ruby-2.6.3
- ruby-2.6.5
- ruby-2.7.1
- ruby-3.2.1
- rust-1.62.1
- rust-1.65.0
- sag-dbtools-1.0
- sage-8.9
- salome-9.10.0
- sbt-1.1.1
- scala-2.12.1
- scala-2.12.4
- scalapack-2.2.0
- SciTE-3.4.0
- scons-2.3.1
- scotch-6.0.0
- scotch-6.1.3
- SDL2-2.0.9
- sep-offprint-1.11
- serf-1.3.6
- serscis-access-modeller-0.16
- sfold-2.2
- sharutils-4.6.3
- shells-1.1
- singularity-2.6.1
- singularity-3.10.4
- singularity-3.4.2
- singularity-3.7.0
- slicer-4.11.20210226
- Smarty-2.6.26
- smpeg-2.0.0
- spacetools-1.0.0
- spacetools-1.1.0
- spacetools-1.2.0
- spack-0.16.0
- spark-2.2.0
- spin-5.2.4
- spin-6.5.1
- spm-12
- sqlite-3.44.2
- sqlite-3.8.4.3
- squashfs-4.3
- StarCCM-12.06.011
- StarCCM-13.06.012
- StarCCM-14.06.013
- StarCCM-15.04.010
- stealfile-1.0
- STSLib-1.0
- subversion-1.14.2
- subversion-1.6.17
- subversion-1.8.9
- subversion-1.9.5
- SuiteSparse-4.2.1
- sundials-6.4.1
- SuperLU-4.3
- SWI-Prolog-7.2.3
- SWI-Prolog-8.4.0
- tcl-8.4.16
- tcl-8.5.14
- tcl-8.6.13
- tcmalloc-1.0
- tcsh-6.18.01
- teams-1.5.00
- tecplot360-2021R2
- tecplot360-2022R1
- tecplot360-2022R2
- tecplot360-2023R1
- TensorFlow-1.15.2
- TensorFlow-1.9
- TensorFlow-2.2.0
- TensorFlow-2.2.1
- TensorFlow-2.4.1
- TensorFlow-r0.7
- texinfo-5.2
- texinfo-7.0.2
- texlive-20220405
- texlive-20230324
- texlive-current
- texstudio-2.12.16
- texstudio-4.5.1
- tgif-4.2.5
- tgrid-5.0.6
- thttpd-2.25b
- thunderbird-102.10.0

- thunderbird-102.11.1
- thunderbird-102.12.0
- thunderbird-115.1.0
- thunderbird-115.10.2
- thunderbird_french-102.10.0
- thunderbird_french-102.11.1
- thunderbird_french-102.12.0
- thunderbird_french-115.1.0
- thunderbird_french-115.10.2
- tidy-5.7.28
- tix-8.1.4
- tk-8.4.16
- tk-8.5.14
- tk-8.6.13
- tmux-3.2a
- tnef-1.4.4
- tomcat-5.5.23
- tomcat-6.0.26
- tomcat-6.0.32
- tomcat-6.0.33
- tomcat-6.0.35
- tomcat-6.0.36
- tomcat-6.0.41
- tomcat-7.0.100
- tomcat-7.0.104
- tomcat-7.0.107
- tomcat-7.0.2
- tomcat-7.0.57
- tomcat-7.0.67
- tomcat-7.0.70
- tomcat-7.0.78
- tomcat-7.0.85
- tomcat-7.0.90
- tomcat-7.0.91
- tomcat7-current
- tomcat-9.0.79
- tomcat9-current
- tomcat_connectors-1.2.42
- tomcat_connectors-1.2.46
- tomcat-current
- transfig-3.2.5
- unafold-3.8
- unrar-3.8.2
- unzip-6.0
- user-utils-1.0
- uudeview-0.5.20
- uuid-1.6.2
- vacation-sendmail-8.12.11
- vacation-sendmail-8.16.1
- valgrind-3.15.0
- verilog-0.9.3
- vmd-1.9.3
- vnc-4.1.3
- VNLB-1.0
- VSCode-1570750623
- web-helpers-1.3
- websphinx-0.5
- weka-3.8.0
- wget-1.16
- wget-1.19
- wget-1.20.3
- wine-1.1.20
- wireshark-2.0.4
- WRF-4.1.2
- wxGTK-2.8.12
- wxGTK-2.8.9
- wxWidgets-3.0.2
- xcal-4.1.18.2
- xcalendar-4.0
- xemacs-21.4.22
- xemacs-21.4.24
- xemacs-21.5.34
- xfig-3.2.5
- xmgr-4.1.2
- xmlsec1-1.2.31
- xpdf-3.04
- xsb-3.3.7
- xv-3.10a
- xz-5.0.0
- xz-5.2.2
- yasm-1.3.0
- zip-3.0
- zlib-1.2.13

D.2 EL9

- a2ps-4.14
- abaqus-2021
- abaqus-2023
- acl-10.1.express
- alpine-2.24
- alpine-2.25
- anaconda3-2023.03
- ansys-2021R1
- ansys-2021R2
- ansys-2022R1
- ansys-2022R2
- ansys-2023R1
- ansys-2023R2
- ant-1.10.11
- ant-1.10.2
- ANTs-2.3.5
- aspell-0.60.8
- bash-4.4
- bazel-0.2.0
- bison-3.7.2
- boost-1.73.0
- buddy-2.4
- camlp5-6.14
- Check-0.15.2
- cmake-3.18.4
- comsol-6.0
- comsol-6.1
- comsol-6.2
- cplex-20.1.0
- cuda-11.5
- cuda-11.8
- cups-2.3.3
- curl-7.86.0

- DbVisualizer-24.1.5
- EasyBuild
- emacs-27.2
- expect-5.45.4
- ffmpeg-4.1.3
- firefox-102.11.0
- firefox-102.12.0
- firefox-102.13.0
- firefox-102.14.0
- firefox-102.15.0
- firefox-102.15.1
- firefox-115.10.0
- firefox-115.2.1
- firefox-115.3.0
- firefox-91.10.0
- firefox-91.11.0
- firefox-91.8.0
- firefox-91.9.0
- firefox-91.9.1
- firefox_french-102.11.0
- firefox_french-102.12.0
- firefox_french-102.13.0
- firefox_french-102.14.0
- firefox_french-102.15.0
- firefox_french-102.15.1
- firefox_french-115.10.0
- firefox_french-115.2.1
- firefox_french-115.3.0
- firefox_french-91.10.0
- firefox_french-91.11.0
- firefox_french-91.8.0
- firefox_french-91.9.0
- firefox_french-91.9.1
- gcc-11.3.0
- gcc-12.2.0
- gcc-4.9.2
- gcc-5.4.0
- gcc-7.3.0
- ghostscript-8.50
- ghostscript-9.50
- gmp-4.3.2
- go-1.12
- go-1.15.6
- go-1.19.3
- gperf-3.0.4
- gurobi-10.0.1
- gurobi-9.1.0
- gv-3.7.4
- httpd-2.4.55
- httpd-2.4.57
- http-parser-2.9.4
- hwloc-2.8.0
- jansson-2.14
- jdk-17
- jdk-17.0.2
- jdk_32b-8u231
- jdk_64b-8u231
- jdk-8
- jdk-8_32b
- json-c-0.16
- libevent-2.1.12
- libjwt-1.15.2
- LibreOffice-7.1.8
- LibreOffice-7.4.7
- libyaml-0.2.5
- lynx-2.8.9
- lz4-1.9.4
- matlab-R2022a
- matlab-R2022b
- matlab-R2023a
- matlab-R2023b
- matlab-R2024a
- matlab-R2024b
- mesa-19.0.3
- modules-3.2.10
- modules-5.3.1
- modules-current
- mpack-1.6
- mpfr-2.4.2
- mpich-4.1.2
- MRtrix-3.0.3
- mysql-5.7.43
- mysql-8.0.31
- nagtools-2.1.10
- nagtools-2.1.3
- nagtools-2.1.4
- nagtools-2.1.5
- nagtools-2.1.6
- nagtools-2.1.7
- nagtools-2.1.8
- nagtools-2.1.9
- nano-6.2
- nasm-2.15.05
- ncurses-6.4
- nmh-1.7.1
- node-v12.18.0
- node-v16.13.0
- nvtop-3.0.1
- ocaml-4.01.0
- OpenFOAM-11.0
- OpenFOAM-12.0
- OpenFOAM-2.4.0
- OpenFOAM-8.0
- OpenFOAM-v2012
- OpenFOAM-v2306
- openmpi-4.1.6
- openpmix-5.0.1
- openssl-1.1.1.current
- openssl-1.1.1n
- openssl-3.0.12
- openssl-3.0.current
- oracle-19c
- ParaView-5.11.2
- perl-5.30.3
- pgadmin4-7.6
- postgresql-12
- postgresql-12.3
- python-2.7.11
- python-3.10.13
- python-3.10.6
- python-3.11.0
- python-3.11.5
- python-3.11.6

- python-3.12.0
- python-3.7.7
- python-3.8.18
- python-3.8.9
- python-3.9.1
- python-3.9.18
- qt-5.14.2
- qt-5.15.10
- qt-5.9
- quota-1.3
- ruby-2.7.1
- singularity-3.10.4
- sqlite-3.44.2
- STSLib-1.0
- tcl-8.4.16
- tcl-8.5.14
- tcl-8.6.13
- tcsh-6.18.01
- tecplot360-2023R1
- texinfo-5.2
- texlive-20220405
- texlive-20230324
- texlive-current
- thunderbird-102.11.1
- thunderbird-102.12.0
- thunderbird-115.1.0
- thunderbird-115.10.2
- thunderbird_french-102.11.1
- thunderbird_french-102.12.0
- thunderbird_french-115.1.0
- thunderbird_french-115.10.2
- tix-8.1.4
- tk-8.4.16
- tk-8.5.14
- tk-8.6.13
- tmux-3.2a
- xemacs-21.4.24

References

[1] 3DS. Abaqus. [online], 2019–2021. https://www.3ds.com/products-services/simulia/products/abaqus/.

[2] ANSYS. FLUENT. [online], 2000–2012. http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/ANSYS+FLUENT.

[3] Belkacem Belabes and Marius Paraschivoiu. Numerical study of the effect of turbulence intensity on VAWT performance. *Energy*, 233:121139, 2021. https://doi.org/10.1016/j.energy.2021.121139.

[4] Belkacem Belabes and Marius Paraschivoiu. CFD study of the aerodynamic performance of a vertical axis wind turbine in the wake of another turbine. In *Proceedings of the CSME International Congress*, 2022. https://doi.org/10.7939/r3-rker-1746.

[5] Belkacem Belabes and Marius Paraschivoiu. CFD modeling of vertical-axis wind turbine wake interaction. *Transactions of the Canadian Society for Mechanical Engineering*, pages 1–10, 2023. https://doi.org/10.1139/tcsme-2022-0149.

[6] Amine Boukhtouta, Nour-Eddine Lakhdari, Serguei A. Mokhov, and Mourad Debbabi. Towards fingerprinting malicious traffic. In *Proceedings of ANT'13*, volume 19, pages 548–555. Elsevier, June 2013.

[7] Amy Brown and Greg Wilson, editors. *The Architecture of Open Source Applications: Elegance, Evolution, and a Few Fearless Hacks*, volume I. aosabook.org, March 2012. Online at http://aosabook.org.

[8] Tariq Daradkeh, Gillian Roper, Carlos Alarcon Meza, and Serguei Mokhov. HPC jobs classification and resource prediction to minimize job failures. In *International Conference on Computer Systems and Technologies 2024 (CompSysTech '24)*, New York, NY, USA, June 2024. ACM.

[9] L. Drummond, H. Banh, N. Ouedraogo, H. Ho, and E. Essel. Effects of nozzle convergence angle on the flow characteristics of a synthetic circular jet in a crossflow. In Bulletin of the American Physical Society, editor, *76th Annual Meeting of the Division of Fluid Dynamics*, November 2023.

[10] Goutam Yelluru Gopal and Maria Amer. Mobile vision transformer-based visual object tracking. In *34th British Machine Vision Conference (BMVC)*, Aberdeen, UK, November 2023. https://arxiv.org/abs/2309.05829 and https://github.com/goutamyg/MVT.

[11] Goutam Yelluru Gopal and Maria Amer. Separable self and mixed attention transformers for efficient object tracking. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, Hawaii, January 2024. https://arxiv.org/abs/2309.03979 and https://github.com/goutamyg/SMAT.

[12] Haotao Lai. An OpenISS framework specialization for deep learning-based person re-identification. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, August 2019. https://spectrum.library.concordia.ca/id/eprint/985788/.

[13] Haotao Lai et al. OpenISS keras-yolo3 v0.1.0, June 2021. https://github.com/OpenISS/openiss-yolov3.

[14] Haotao Lai et al. Openiss person re-identification baseline v0.1.1, June 2021. https://github.com/OpenISS/openiss-reid-tfk.

[15] MathWorks. MATLAB. [online], 2000–2012. http://www.mathworks.com/products/matlab/.

[16] Serguei Mokhov, Jonathan Llewellyn, Carlos Alarcon Meza, Tariq Daradkeh, and Gillian Roper. The use of containers in OpenGL, ML and HPC for teaching and research support. In *ACM SIGGRAPH 2023 Posters*, SIGGRAPH '23, New York, NY, USA, 2023. ACM. https://doi.org/10.1145/3588028.3603676.

[17] Serguei A. Mokhov. The use of machine learning with signal- and NLP processing of source code to fingerprint, detect, and classify vulnerabilities and weaknesses with MARFCAT. Technical Report NIST SP 500-283, NIST, October 2011. Report: http://www.nist.gov/

manuscript-publication-search.cfm?pub_id=909407, online e-print at http://arxiv.org/abs/1010.2511.

[18] Serguei A. Mokhov. *Intensional Cyberforensics*. PhD thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, September 2013. Online at http://arxiv.org/abs/1312.0466.

[19] Serguei A. Mokhov, Michael J. Assels, Joey Paquet, and Mourad Debbabi. Automating MAC spoofer evidence gathering and encoding for investigations. In Frederic Cuppens et al., editors, *Proceedings of The 7th International Symposium on Foundations & Practice of Security (FPS'14)*, LNCS 8930, pages 168–183. Springer, November 2014. Full paper.

[20] Serguei A. Mokhov, Michael J. Assels, Joey Paquet, and Mourad Debbabi. Toward automated MAC spoofer investigations. In *Proceedings of C3S2E'14*, pages 179–184. ACM, August 2014. Short paper.

[21] Serguei A. Mokhov, Joey Paquet, and Mourad Debbabi. The use of NLP techniques in static code analysis to detect weaknesses and vulnerabilities. In Maria Sokolova and Peter van Beek, editors, *Proceedings of Canadian Conference on AI'14*, volume 8436 of *LNAI*, pages 326–332. Springer, May 2014. Short paper.

[22] Parna Niksirat, Adriana Daca, and Krzysztof Skonieczny. The effects of reduced-gravity on planetary rover mobility. *International Journal of Robotics Research*, 39(7):797–811, 2020. https://doi.org/10.1177/0278364920913945.

[23] N. Ouedraogo, A. Cyrus, and E. Essel. Effects of Reynolds number on the wake characteristics of a Notchback Ahmed body. In Bulletin of the American Physical Society, editor, *76th Annual Meeting of the Division of Fluid Dynamics*, November 2023.

[24] Newton F. Ouedraogo and Ebenezer E. Essel. Unsteady wake interference of unequal-height tandem cylinders mounted in a turbulent boundary layer. *Journal of Fluid Mechanics*, 977:A52, 2023. https://doi.org/10.1017/jfm.2023.952.

[25] Newton F. Ouedraogo and Ebenezer E. Essel. Effects of Reynolds number on the wake characteristics of a Notchback Ahmed body. *Journal of Fluids Engineering*, 146(11):111302, 05 2024.

[26] Chet Ramey. The Bourne-Again Shell. In Brown and Wilson [7]. http://aosabook.org/en/bash.html.

[27] Farshad Rezaei and Marius Paraschivoiu. Placing a small-scale vertical axis wind turbine on roof-top corner of a building. In *Proceedings of the CSME International Congress*, June 2022. https://doi.org/10.7939/r3-j7v7-m909.

[28] Farshad Rezaei and Marius Paraschivoiu. Computational challenges of simulating vertical axis wind turbine on the roof-top corner of a building. *Progress in Canadian Mechanical Engineering*, 6, 1–6 2023. http://hdl.handle.net/11143/20861.

[29] Rob Schreiber. MATLAB. *Scholarpedia*, 2(6):2929, 2007. http://www.scholarpedia.org/article/MATLAB.

[30] The MARF Research and Development Group. The Modular Audio Recognition Framework and its Applications. [online], 2002–2014. http://marf.sf.net and http://arxiv.org/abs/0905.1235, last viewed May 2015.