August 2023

Get your code/software to run on ARC systems

Matthew Brown
Computational Scientist
Advanced Research Computing
Virginia Tech

VIRGINIA TECH

Fall 2023 ARC workshop Series

Introductory / Orientation Workshops (75 minutes each):

- *Advanced Research Computing (ARC) Overview*
- *Connect to ARC systems and run your first jobs*
- *Get your software/code to run on ARC*

Series 1: 8/17 - 8/18
Series 2: Wednesday afternoons: 9/27, 10/4, 10/11

Special Topics (75 minutes each):

- *Monitoring Resource Utilization and Job Efficiency*
- *Getting the Best Data Storage Performance on ARC Filesystems*
- *Launching in Parallel*

Series 1: Biweekly, Tuesday mornings 9/5, 9/19, 10/3
Series 2: Weekly, Thursday afternoons 10/26, 11/2, 11/9

VIRGINIA TECH

Expectations

- This is an informal workshop
- Mostly informational about ARC and research computing at VT
- I want to hear your questions
- Welcome to use chat to ask questions + some time at the end
- Feedback needed to help improve future workshops
    - One up / one down at the end

VIRGINIA TECH

Get your software/code to work on ARC systems

Description

ARC systems run software which spans the full spectrum of modern research computing. This workshop addresses several of the most common software delivery models and how they can be accessed and used on ARC systems.

The demonstrations will be predominantly via the Linux shell command line interface and will cover our "software modules" system, python environments via Anaconda, and also the main components needed for building software from source codes, particularly MPI software. Interactive shell jobs are an important tool for each of these models and will be used to demonstrate how researchers can begin to set up their own software environments and develop workflows on ARC systems.

Participants who already have ARC accounts are invited to follow along with the demonstrations if desired.

Outline:
- Interactive shell jobs
- Search for, load, and manage modules
- Python with Anaconda environments
- Building from source

First – a bit about file transfers

Command line tools: scp, rsync, wget, curl, tar

Graphical tools: ~~MobaXterm, FileZilla, WinSCP~~

Other organizations: Globus, cloud providers

For speed, package and compress files to optimize transfers!

https://www.docs.arc.vt.edu/usage/scratch.html

VIRGINIA TECH

Interactive shell jobs

Why not just use the login nodes?

1. Login nodes are not always identical to compute nodes. Software build/configuration environment should mirror the target execution environment.
2. Login nodes are a shared resource. Avoid running anything intensive on the login nodes.
3. The CPU and memory available on login nodes is limited to minimize the impact of accidental abuse.

Interactive shell jobs

Simplest option: `interact --account=<myacct>` (default settings apply)

Customizable: Any `sbatch/salloc/srun` options can be used to modify the resource request:

```
--partition=   --nodes=   --ntasks-per-node=   --cpus-per-task=
--gres=gpu:1   --time=d-hh:mm:ss
```

Advanced option: Compose your own `salloc` or `srun` commands.

`srun [options] <cmd>`	get allocation, run `cmd` and quit
`salloc -> srun`	get allocation -> run command in allocation
`srun --pty [options] bash`	get allocation and shell on allocated node

Software modules

ARC supplied packages on ARC systems: Easybuild + modules

Module commands:

```
module list
module reset
module avail
module spider <string>
```

Standalone modules, toolchains, dependencies:

```
module load intel
module list
module load ParaView
```

Software modules

What actually happens when I load a module?

Loading a module sets environment variables which customize the current runtime environment to make a package work.

Commands to investigate:

```
env | grep <str>
which <program>
echo $PATH; echo $LD_LIBRARY_PATH
ldd <path to binary file>
module show <modulename>
```

https://www.docs.arc.vt.edu/usage/video.html

Easybuild

What is it? An open source, HPC community supported package installation and management system. Build from source whenever available.

ARC uses Easybuild to provide packages system-wide. You can use it to install packages, available as modules just for yourself.

Usage overview: (must be on node-type where you intend to use software)

```
module load EasyBuild
eb -S <search string>              eb -D <package name> (optional)
eb <package name>                  eb --help | less
```

VIRGINIA TECH

Anaconda environments for python

Node type matters: You can only expect to use an environment on the partition where it was built. Environments are not portable.

- Plan ahead to organize your environments. `~/env/tc/a100/tf2.8`
- Use `/projects` to share environments with a group.

Avoid ~~`conda init`~~ and ~~`conda activate`~~:

One HOME directory is shared across all ARC systems which have many different, unique, Anaconda installations.

- `source activate /path/to/env`

Anaconda environments for python

Standard steps:

Get interactive job on a node of the correct type	`interact --partition=dgx_dev_q -gres=gpu:1 --account=personal --time=0:10:00`

Load an Anaconda module and other needed modules	`module reset`
	`module load Anaconda3`
	`(optional) module load foss`

Create an environment <u>at a path</u>	`conda create -p ~/env/tcdgx/tf`
Activate the environment	`source activate ~/env/tcdgx/tf`
Install packages into the environment	`conda install python=3.9 tensorflow`

VIRGINIA TECH

Anaconda environments for python

Other info to know:

1. Miniconda is smaller, uses same package manager (`conda`)
2. Create environments once, not repeatedly in batch jobs
3. For intensive program/library I/O, consider `/globalscratch` or `$TMPFS`
4. Environments can be loaded in OnDemand Jupyter notebooks
5. Specify only the most essential packages/versions when installing and let `conda` do the rest.

Containers (Apptainer)

Containerization overview:

- (Usually) Lightweight overlay on a host OS
- Often considered to be "highly portable" or "self contained" (caveats may apply)
- module spider apptainer
- Run command in container: `apptainer exec [options] ctnr.sif command`
- Docker containers can be set up on your personal computer, then pushed to Dockerhub
- Apptainer can pull from Dockerhub and convert Docker containers to `.sif`
- Rapidly growing availability of containerized applications
- `-B<dir>` to "bind mount" directories into the container (eg. `/projects/myproj`)
- -gpu?

VIRGINIA TECH

Linux software in a nutshell

Overview:

Architectures, compilers, libraries, and linking

- All ARC systems currently are `x86_64`, but have had `ppcle`. `arm64` future?
- Chip capabilities (`lscpu`, wikichip.org): vectorizations: `AVX`, `AVX2`, `AVX512`
- Memory subsystem organization (`numactl --hardware`)
- GCC, Intel, AOCC , PGI -> NVHPC, Cray
- MPI, OpenMP, BLAS,

Free open source software - FOSS

Distribution-based package management (root only): rpm, yum, apt, debs

Building Software from Source

The tools are all available: `make`, `cmake`, compilers, libraries, etc.
Use EasyBuild to satisfy as many dependencies as you can.

Typical sequence:

```
module reset
module load foss
tar xf newpackage_4.1.tar.gz
cd newpackage_4.1
./configure --prefix=/projects/brownlab/software/newpackage_4.1/
make
make install
```

Message Passing Interface (MPI)

Standardized interface for inter-process communication.

Several implementations exist (MPICH, OpenMPI, MVAPICH, Intel MPI, IBM), but each includes

- compiler wrappers (eg. mpicc, mpif90)
- header files and libraries
- can be built to be aware of specialized networks and resource managers (Slurm, Infiniband, etc)

To use:

- write code that calls MPI functions/routines
- compile and link with MPI compiler and libraries
- launch programs with mpirun or srun inside a job allocation

VIRGINIA TECH

Message Passing Interface (MPI)

https://github.com/AdvancedResearchComputing/examples/tree/master/mpi

```
mpi_quad.c

module load foss/2020b
mpicc mpi_quad.c -o tc_nq_f20b_mpiquad
salloc --nodes=4 --ntasks-per-node=16 --time=0:02:00 --account=personal
mpirun ./tc_nq_f20b_mpiquad
```

Compilation of programs may need additional
- paths header files (-I/path/to/inc)
- paths to libraries (-L/path/to/lib)
- library names (-ldepend for /path/to/lib/libdepend.so)

Intel, GCC, NVHPC, etc. all use different options which are not cross compatible. Use manual and --help to investigate. Edit makefiles to customize for ARC software/versions.

VIRGINIA TECH

Thanks for watching and listening!

ARC Website:	www.arc.vt.edu
My contact info:	Matthew Brown brownm12@vt.edu

VIRGINIA TECH