April 2022

Get your code/software to run on ARC systems

Matthew Brown, Computational Scientist
Advanced Research Computing
Information Technology
Virginia Tech

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

Spring 2022 ARC workshop Series

April 12th or 13th	Advanced Research Computing (ARC) Overview	Mission and goals, resources and services, getting started, getting assistance
April 19th or 20th	Connect to ARC systems and run your first jobs	VPN, Windows Subsystem for Linux, Git/BASH, MobaXterm/PuTTY, OnDemand, ssh keys, screen/tmux
April 26th or 27th	Get your software to run on ARC	File management, finding things, monitoring utilization, understanding your environment, loading software
May 12th – 19th	Software Carpentries (VT Libraries)	Foundations of Unix, Git and Python. Programming with Python. R for Reproducible Research. The Unix Shell. Version Control with Git

VIRGINIA TECH

Get you software/code to work on ARC systems

Description

ARC systems run software which spans the full spectrum of modern research computing. Many fields have evolved their software in various ways, but most often within the support models of research computing centers like ARC. 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 indispensable resource 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. Attendees 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

VIRGINIA TECH

First – a bit about file transfers

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

Graphical tools: MobaXterm, FileZilla, WinSCP

Other organizations: Globus (VT not currently licensed), cloud

Package and compress files to optimize transfers!

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

VIRGINIA TECH

Interactive shell jobs

Reminder: Please try not to abuse login nodes. Node type matters!

Simplest option: `interact --account=<myacct>` (accept defaults)

Customizable: All `sbatch`/`salloc`/`srun` options should be accepted.

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

VIRGINIA TECH

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
```

VIRGINIA TECH

Software modules

What actually happens when I load a module?

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

Commands to investigate:

```
env | grep <str>
echo $PATH; echo $LD_LIBRARY_PATH
module show <modulename>
```

```
which <program>
ldd <path to binary file>
```

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

VIRGINIA TECH

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 <package name>
eb -D <package name> (optional)
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`

VIRGINIA TECH

Anaconda environments for python

Standard steps:

1. Get interactive job on a node of the correct type
2. Load an Anaconda module and other needed modules (eg. cuda)
3. Create an environment at a path: `conda create -p ~/env/tca100/tf`
4. Activate the environment: `source activate ~/env/tca100/tf28`
5. 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 /fastscratch 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 (Singularity)

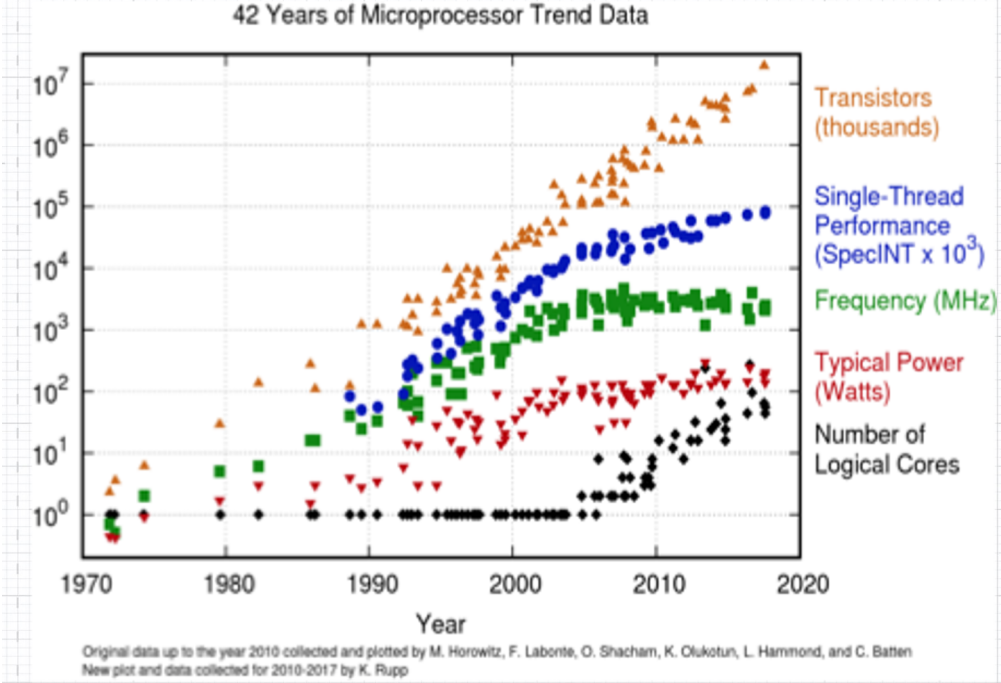
Containers overview:

- (Usually) Lightweight overlay on a host OS
- Often considered to be "highly portable" or "self contained" (caveats may apply)
- `module spider singularity`
- Run `command` in container: `singularity exec [options] ctnr.sif command`
- Docker containers can be set up on your personal computer, then pushed to Dockerhub
- Singularity 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`)

VIRGINIA TECH

Parallelism is the New Moore's Law

- Power and energy efficiency impose a key constraint on design of micro-architectures
- Clock speeds have plateaued
- Hardware parallelism is increasing rapidly to make up the difference



VIRGINIA TECH

"Pleasingly Parallel"

Computations are independent and can be executed simultaneously.

Examples: Parameter sweeps, numerical integration, BLAST searches

Parallelization approaches and tools:

- At BASH script level: GNU/`parallel`, `srun`
- Matlab "`parfor`" to replace certain "`for`" loops
- FORTRAN/C codes on data structure operations: OpenMP (threads) and/or MPI (ranks/tasks/processes)

GNU/parallel

Multifunctional utility with lots of features and usages. Great for passing arguments to repeated commands.

Much better for load balancing than BASH for loops. Manual has lots of examples (`man parallel`)

Examples:

1. Pass sequence of parameters to parallel executed code:
   ```
   ls -d mw* | parallel tar -czf {}.tar.gz {}
   seq 10 | parallel 'sleep $((5+2*{}))' &
   ```
2. `parallel` can load balance (-j<#>): `seq 10 | parallel -j3 'sleep $((5+2*{}))' &`
3. `parallel` + `srun` when running on several machines provides complimentary features of `srun`

`srun` features:

knows about hosts allocated to job and requested task layout

can control cpu-binding

VIRGINIA TECH

Linux software in a nutshell

Overview:

- Architectures, compilers, libraries, and linking
  - Huckleberry – `ppcle`. All others - x86_64
  - Chip capabilities (`lscpu`, wikichip.org): vectorizations: `AVX, AVX2, AVX512`
  - GCC, Intel, AOCC , PGI -> NVHPC, Cray
  - MPI, OpenMP, BLAS,
- Free open source software - FOSS
- Distribution-based package management (root only): `rpm, yum, apt, debs`

Performance Monitoring

From login node:

`jobload <jobid>` (while job is running)

`seff <jobid>` (when job has completed)

You may SSH to nodes on which you have allocated resources. Then

- `htop -u <username>` (process list, cpu-load per core, memory usage)
- `nvidia-smi` (GPU status)
- `gpumon`

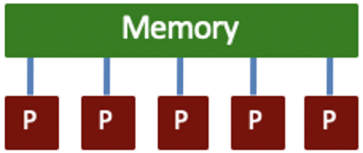
CPU to task binding:

`--cpu-bind=verbose`

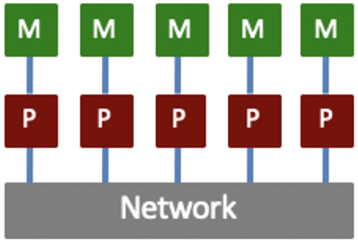
add column for `PROCESSOR` to htop display

VIRGINIA TECH

Shared vs. Distributed Memory

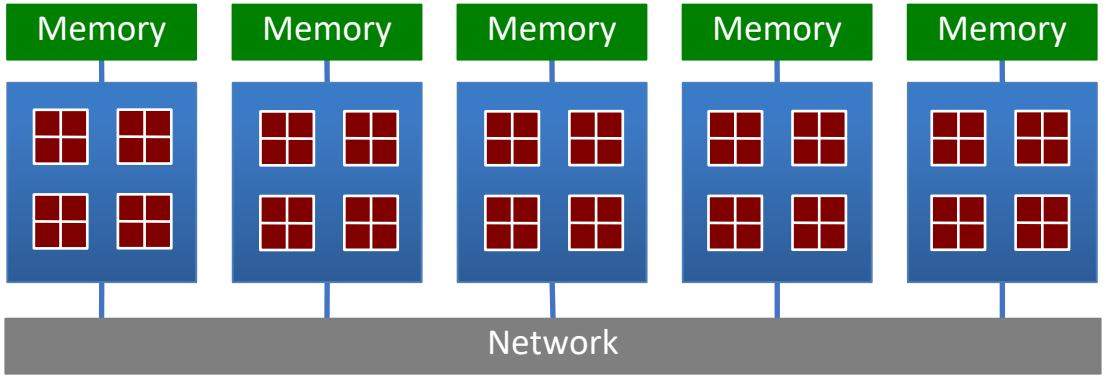


- All processors have access to a pool of shared memory
- Access times vary from CPU to CPU in NUMA systems
- Example: CPUs on same node
- OpenMP



- Memory is local to each processor
- Data exchange by message passing over a network
- Example: Clusters with single-socket blades
- MPI

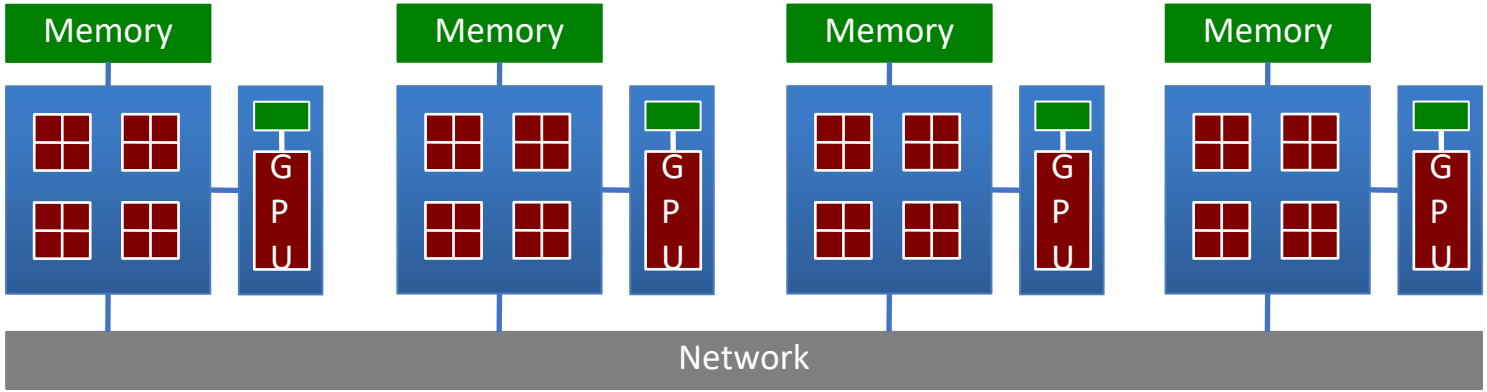
Multi-Core Systems



- Current processors many cores on a die
- Communication details are increasingly complex
  - Cache access (shared L3)
  - Main memory access (cores share memory channels)
  - Quick Path / Hyper Transport socket connections
  - Node to node connection via network

```
numactl --hardware
```

Accelerator-Based Systems



- Calculations made in both CPUs and GPUs
- No longer limited to single precision calculations
- Load balancing critical for performance
- Requires specific libraries and compilers (CUDA, OpenCL)

"Built-in" or library based parallelism

MATLAB examples:

1. Built-in Arithmetic Uses Available Cores

```
module load module tinkercliffs-rome/matlab/R2021a
```

```
srun -A personal --nodes=1 --ntasks=1 --cpus-per-task=32 --pty matlab -nosplash -nosoftwareopengl -sd `pwd`
```

```
>> N=25000; A=rand(N); B=rand(N); tic; A*B; toc;
```

2. `parpool` spawns workers to which `parfor` can farm out tasks

```
>> parpool(32);
```

```
>> tic; n=200; A=2000; a=zeros(1,n); parfor i=1:n; a(i)=max(abs(eig(rand(A)))); end; toc;
```

3. Using GPUs: `gpuArray`

VIRGINIA TECH

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

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.

Support, Consultation and Collaboration

ARC Helpdesk: https://arc.vt.edu/support

ARC Helpdesk GRAs work as a team to handle most incoming questions/problems.

"How do I setup SSH keys for authentication?" *"What can I do to get my job to launch faster?"* *"Why did my job stop?"*

"Is MATLAB available on Huckleberry?" *"How can I share my files with my collaborator?"*

Escalate to ARC Computational Scientists as needed.

Office Hours (https://arc.vt.edu/office-hours)

VIRGINIA TECH

Thanks for watching and listening!

ARC Website: www.arc.vt.edu

My contact info: Matthew Brown
brownm12@vt.edu

VIRGINIA TECH