Software Install and Containers on ARCHER

Adrian Jackson

a.jackson@epcc.ed.ac.uk

Reusing this material

Partners

UKRI

Engineering and Physical Sciences Research Council

Natural Environment Research Council

epcc

THE UNIVERSITY of EDINBURGH

Hewlett Packard Enterprise

Installing software on ARCHER2

- ARCHER2 has a lot of software already installed
  - Using `module` system

  ```
  module avail
  module list
  module load PrgEnv-gnu
  ```

  - Optimised central installs are preferable

  ```
  module load cray-python
  ```

    Currently version 3.8.5.0

    Contains numpy, scipy, mpi4py, dask

  ```
  module load cray-R
  ```

    Currently version 4.0.3.0

- Two challenges to installing software
  - Cray Linux may be non-standard, and may not have dependencies installed
  - Default `home` directory installs won't work for the compute nodes

Scientific libraries

- Cray libsci loaded by default:

```
module list
module show cray-libsci
```

- Provides optimised:
  - blas
  - lapack
  - scalapack
- More info available on the system:

```
man intro_libsci
man intro_blas1   intro_blas2   intro_blas3
man intro_lapack
man intro_scalapack
```

- Other scientific software packages pre-installed

```
    module avail
```

  - FFTW, HDF5, NetCDF, ADIOS
  - ARPACK, Boost, Eigen, GLM, HYPRE, Matio, Intel MKL, MUMPS, PETSc, SLEPc, Trilinos, SuperLU/SuperLU_DIST
  - Metis/Parmetis, Scotch/PT-Scotch

Compilers

- Three different compiler environments available on ARCHER2:
  - AMD Compiler Collection (AOCC)
    `module load PrgEnv-aocc`
    - Based on clang and flang
  - GNU Compiler Collection (GCC)
    `module load PrgEnv-gnu`
  - HPE Cray Compiler Collection (CCE) (default)
    `module load PrgEnv-cray`
    - Cray Fortran compiler and clang for C/C++
- Compilation undertaken using:
  - `cc, CC, ftn`
- Different compiler versions are also available
  - i.e. `module swap gcc gcc/11.2.0`

```
module avail cce
module avail gcc
module avail aocc
```

Installing software

- Installing your own software for use on the compute nodes
  - Remember only `/work` is available on the compute nodes
- Python

```
module load cray-python
export PYTHONUSERBASE=/work/t01/t01/auser/.local
```

  - You will need to change `t01` to the project code for your project, and `auser` to your username

```
export PATH=$PYTHONUSERBASE/bin:$PATH
```

  - Can use virtual environments, i.e.:

```
source <<path to virtual environment>>/bin/activate
```

  - Can then use pip or conda to install software

```
pip install --user dask distributed
```

  - `--user` isn't required if virtual environments are being used
- Some python installs may need flags set for compilation

```
export CC=cc
export CXX=CC
export FC=ftn
```

- BLAS, LAPACK libraries provided from the `cray-libsci` module (loaded by default)

Installing software

- R
  ```
  export R_LIBS_USER=/work/z19/z19/adrianj/Rinstall
  R
  install.packages('snow’)
  ```
- May also need to configure install environment:
  - Create a preference directory
    ```
    mkdir ~/.R
    ```
  - Add this to:
    ```
    ~/.R/Makevars
    ```
  - With the following lines:
    ```
    CC = cc
    CXX = CC
    FC = ftn
    ```
  - Then can install from the command line:
    ```
    R CMD INSTALL Rmpi_0.6-9.2.tar.gz --configure-args=" --with-Rmpi-
    type=CRAY"
    ```

R

- Packages in library ‘/opt/R/4.0.3.0/lib64/R/library’:

base	The R Base Package
boot	Bootstrap Functions (Originally by Angelo Canty for S)
class	Functions for Classification
cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.
codetools	Code Analysis Tools for R
compiler	The R Compiler Package
datasets	The R Datasets Package
foreign	Read Data Stored by 'Minitab', 'S', 'SAS’, 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...
graphics	The R Graphics Package
grDevices	The R Graphics Devices and Support for Colours and Fonts
grid	The Grid Graphics Package
KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)
lattice	Trellis Graphics for R
MASS	Support Functions and Datasets for Venables and Ripley's MASS
Matrix	Sparse and Dense Matrix Classes and Methods
methods	Formal Methods and Classes
mgcv	Mixed GAM Computation Vehicle with Automatic Smoothness Estimation
nlme	Linear and Nonlinear Mixed Effects Models
nnet	Feed-Forward Neural Networks and Multinomial Log-Linear Models
parallel	Support for Parallel computation in R
rpart	Recursive Partitioning and Regression Trees
spatial	Functions for Kriging and Point Pattern Analysis
splines	Regression Spline Functions and Classes
stats	The R Stats Package
stats4	Statistical Functions using S4 Classes
survival	Survival Analysis
tcltk	Tcl/Tk Interface
tools	Tools for Package Development
utils	The R Utils Package

```
Rscript -e "installed.packages()"
```

Containers

- Containers allow separation of kernel and user space for operating systems and applications
  - Enable different user space configurations for a given kernel space
  - Interface level between the two
  - Virtualises the hardware and operating system from the user software perspective
  - Lighter weight than full virtualisations (VM) but less isolated
- Docker is an example
  - Docker images are the files and directories that a docker container will be created from
  - Container is a runtime image (lightweight virtual machine)
  - Docker im ages can be obtained from the Docker hub https://hub.docker.com/

Singularity

- Docker has some security configuration issues that restrict its use on shared resources like ARCHER2
- HPC systems also have specific requirements around optimised filesystems and networks
  - Singularity is a container implementation designed for HPC systems
  - Others are also available (shifter, Charliecloud, etc.…)
- Singularity has two different versions
  - https://apptainer.org/
  - https://sylabs.io/
- Singularity on ARCHER2

```
singularity --version
```

Using singularity

- Download and run a docker container

```
singularity pull python-3.9.9.sif docker://python:3.9.9-slim-buster
singularity run python-3.9.9.sif python -c "print(‘hello’)”
```

- Can also get containers from http://datasets.datalad.org/?dir=/shub

```
singularity pull hello-world.sif shub://vsoch/hello-world
singularity run hello-world.sif
```

- Different ways to run things using singularity

```
singularity run hello-world.sif
singularity shell hello-world.sif
singularity exec python-3.9.9.sif python -c “print(‘hello’)”
```

- Check the default command/execution

```
singularity inspect -r hello-world.sif
```

- Some containers also available on ARCHER2

```
/work/y07/shared/singularity-images
```

Singularity users and files

- By default singularity (on ARCHER2) will bring in your user and groups from the host system

```
adrianj@ln04:~> singularity shell -B /work/z19/z19/adrianj  hello-world.sif
Singularity> whoami
adrianj
Singularity> groups
z19 archer2-tds-login archer2-4c-login castep-admin cse-admin archer2-login
```

  - Enables file access and permissions etc… to be maintained
- Filesystem is not imported…
  - …except your home directory

```
adrianj@ln04:~> singularity shell hello-world.sif
Singularity> pwd
/home/z19/z19/adrianj
```

- Can bring in additional directories using –B flag

```
adrianj@ln04:~> singularity shell -B /work/z19/z19/adrianj  hello-world.sif
```

  - Can specify bind path:

```
adrianj@ln04:~> singularity shell -B /work/z19/z19/adrianj:/workdir  hello-world.sif
```

- Container is read only except your home directory and bind directories

Parallel applications using Singularity

- Node local (shared memory) applications
  - Singularity usage model the same as normal applications

```
#!/bin/bash --login
#SBATCH --job-name=my_app
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=8
#SBATCH --time=00:10:00

#SBATCH --account=[budget code]
#SBATCH --partition=standard
#SBATCH --qos=standard
export OMP_NUM_THREADS=8
singularity run $SLURM_SUBMIT_DIR/my_app.sif
```

- Distributed memory applications (MPI) requires more care
  - Need MPI compatibility between host and container

Parallel applications using Singularity

```
#!/bin/bash
#SBATCH --job-name=singularity_parallel
#SBATCH --time=0:10:0
#SBATCH --nodes=2
#SBATCH --tasks-per-node=128
#SBATCH --cpus-per-task=1
#SBATCH --partition=standard
#SBATCH --qos=standard
#SBATCH --account=[budget code]
# Set the number of threads to 1.
# This prevents any threaded system libraries from automatically using threading.
export OMP_NUM_THREADS=1
# Set the LD_LIBRARY_PATH environment variable within the Singularity container
# to ensure that it used the correct MPI libraries.
export SINGULARITYENV_LD_LIBRARY_PATH= \
    /opt/cray/pe/mpich/8.1.9/ofi/gnu/9.1/lib-abi-mpich: \
    /opt/cray/pe/pmi/6.0.13/lib: \
    /opt/cray/libfabric/1.11.0.4.71/lib64: \
    /usr/lib64/host: \
    /usr/lib/x86_64-linux-gnu/libibverbs: \
    /.singularity.d/libs
# Set the options for the Singularity executable.
# This makes sure Cray Slingshot interconnect libraries are available
# from inside the container.
BIND_OPTS="-B /opt/cray,/usr/lib64:/usr/lib64/host,/usr/lib64/tcl"
BIND_OPTS="${BIND_OPTS},/var/spool/slurmd/mpi_cray_shasta"
# Launch the parallel job.
srun --hint=nomultithread --distribution=block:block \
    singularity run ${BIND_OPTS} osu_benchmarks.sif \
        collective/osu_allreduce
```

Parallel applications using Singularity

- Interactive compile

```
singularity run  -B /work/z19/z19/adrianj:/workdir
/work/y07/shared/singularity-images/mpich_base.sif mpicc -fopenmp
-o /workdir/mpi_hello_world /workdir/mpi_hello_world.c
```

- Interactive run

```
srun --hint=nomultithread --distribution=block:block --nodes=1 --
tasks-per-node=16 --cpus-per-task=8 --exclusive --
partition=standard --qos=short --reservation=shortqos --
account=z19 --time=0:20:0 singularity run  "-B
/work/z19/z19/adrianj:/workdir,/work/y07/shared,/opt:/opt,/usr/lib
64:/usr/lib64/host,/usr/lib64/tcl,/var/spool/slurmd/mpi_cray_shast
a" /work/y07/shared/singularity-images/mpich_base.sif
/workdir/mpi_hello_world
```

MPI in Singularity

- Different modes for using MPI inside a Singularity container
  - https://apptainer.org/user-docs/3.7/mpi.html#singularity-and-mpi-applications
  - Host mode
    - Use the host MPI to run Singularity
    - Enables integration with batch system
    - Needs compatible MPI between host and container
    - Needs container to be configured to use high performance network
  - Bind mode
    - No MPI required within the container
    - Package application into the container
    - Specify where the host MPI is installed so can be accessed within the container

MPI in Singularity

- Host mode example:
    - Build definition

```
Bootstrap: docker
From: ubuntu:20.04

%files
    /home/singularity/osu-micro-benchmarks-5.8.tgz /root/
    /home/singularity/mpich-3.4.3.tar.gz /root/

%environment
    export SINGULARITY_MPICH_DIR=/usr
    export OSU_DIR=/usr/local/osu/libexec/osu-micro-benchmarks/mpi

%post
    apt-get -y update && DEBIAN_FRONTEND=noninteractive apt-get -y install build-essential libfabric-dev libibverbs-dev gfortran
    cd /root
    tar zxvf mpich-3.4.3.tar.gz && cd mpich-3.4.3
    echo "Configuring and building MPICH..."
    ./configure --prefix=/usr --with-device=ch4:ofi && make -j8 && make install
    cd /root
    tar zxvf osu-micro-benchmarks-5.8.tgz
    cd osu-micro-benchmarks-5.8/
    echo "Configuring and building OSU Micro-Benchmarks..."
    ./configure --prefix=/usr/local/osu CC=/usr/bin/mpicc CXX=/usr/bin/mpicxx
    make -j6 && make install

%runscript
    echo "Rank ${SLURM_PROCID} - About to run: ${OSU_DIR}/$*"
    exec ${OSU_DIR}/$*
```

    - Build command

```
singularity build osu_benchmarks.sif osu_benchmarks.def
```

Creating images

- To modify images/build new images need
  - Root permissions on a system with singularity installed
  - Docker installed on a system (using a docker singularity container)
- Create image definition file

```
Bootstrap: docker
From: ubuntu:20.04

%post
    apt-get -y update && apt-get install -y python3

%runscript
    python3 -c 'import sys; print("Hello World! Hello from Python %s.%s.%s in our custom Singularity image!" % sys.version_info[:3])'
```

- Build image

```
singularity build my_test_image.sif my_test_image.def
```

  - Or

```
docker run --privileged --rm --mount type=bind,source=${PWD},target=/home/singularity quay.io/singularity/singularity:v3.7.3-slim build /home/singularity/my_test_image.sif /home/singularity/my_test_image.def
```

- Can make more complex/functional images
  - Different sections for definition files:
    - %setup, %files, %environment, %startscript, %test, %labels, %help
    - https://apptainer.org/user-docs/3.7/definition_files.html#sections
  - Can sign containers for distributed etc…

Summary

- Plenty of software already available on ARCHER2
- Plus a range of compilers
- Installing your own R and Python libraries is straightforward
  - But ensuring they’re available on the compute nodes requires configuration
- More complex software installs may benefit from containers
  - Basic container functionality simple
  - Interfacing with MPI and the /work filesystems require more care
- Ensuring software is as efficient as possible is important if using large amounts of compute time
  - Placement and binding for threads/processes important
  - Optimised maths libraries
  - New compilers and optimised compile options
  - etc…