ARCHER2 for Data Scientists

Practical 3: Software installs and containers

Adrian Jackson

1 Introduction

In this exercise we are going to try install Python and R packages on ARCHER2 and experiment with Singularity containers.

2 Python installs

Recall that there is an optimised Python module available on ARCHER2:

```
module load cray-python
```

Once that is loaded, to install your own extra software you need to do the following:

```
export PYTHONUSERBASE=/work/ta144/ta144/$USER/.local
export PATH=$PYTHONUSERBASE/bin:$PATH
```

Then you can install your own software. Try installing the following:

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

And then testing it:

```
python -c "import dask.distributed"
```

Try also testing it on the compute nodes using an interactive srun session, i.e.:

```
srun --nodes=1 --tasks-per-node=1 --cpus-per-task=1 --exclusive --
partition=standard --qos=short --time=0:20:0 python -c "import
dask.distributed"
```

Does it work for you?

3 R installs

Recall that there is an optimised R module available on ARCHER2:

```
module load cray-R
```

Once that is loaded, to install your own extra software you need to do the following:

```
export R_LIBS_USER=/work/ta144/ta144/$USER/Rinstall
```

Try installing snow using these commands:

```
R

install.packages('snow’)
```

You can choose whichever mirror you like. Once it has finished you can quit R (using `q()` or `ctrl-d`).

Now test that it has been installed:

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

And also check you can access it on the compute nodes on ARCHER2:

```
srun   --nodes=1   --tasks-per-node=1   --cpus-per-task=1   --exclusive   --
partition=standard         --qos=short     --time=0:20:0     Rscript     -e
"installed.packages()"
```

Does that work?

4 Containers

Singularity is the container infrastructure we use on ARCHER2. Try the following to check you can get and run a container on the system:

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

singularity run hello-world.sif
```

This should print out:

```
RaawwWWWWWRRRR!! Avocado!
```

Now you are able to run a container, we can try using a container to run some of our own software. Use the container provided at `/work/y07/shared/singularity-images/mpich_base.sif` , which is a container with the MPICH MPI library installed, although not the Cray version of the library, compile the following MPI code:

```
#include <stdio.h>
#include <mpi.h>
#include <omp.h>
#include <unistd.h>
#include <string.h>
int main(int argc, char *argv[])
{
 int rank, thread;
 char hnbuf[64];
 MPI_Init(&argc, &argv);
 MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  memset(hnbuf, 0, sizeof(hnbuf));
 (void)gethostname(hnbuf, sizeof(hnbuf));
 #pragma omp parallel private(thread)
 {
   thread = omp_get_thread_num();
   #pragma omp barrier
   printf("Hello from rank %d, thread %d, on %s\n", rank, thread, hnbuf);
 }
 MPI_Finalize();
 return(0);
}
```

To build this using the compiler in the container, you will first need to copy the above code and save it to a file on ARCHER2. Then you can run a command like this to build the executable (assuming you saved the above code into a file called `mycode.c`:

```
singularity exec -B`pwd`:`pwd` /work/y07/shared/singularity-
images/mpich_base.sif mpicc -o mycode -fopenmp mycode.c
```

This should then create and executable you can run this MPI application using Singularity like this:

```
export OMP_NUM_THREADS=2
singularity run  "-B
/work/y07/shared,/opt:/opt,/usr/lib64:/usr/lib64/host,/usr/lib64/tcl"
/work/y07/shared/singularity-images/mpich_base.sif mpirun -n 2  ./mycode
```

This runs the container to start the MPI application, using 2 MPI processes and 2 OpenMP threads per MPI process. The -B options bind host filesystem paths to the container so that the MPI library and other libraries linked into the application are available at runtime.

The methods outlined above should enable you to use applications inside containers on the login nodes on ARCHER2. To use Singularity containers on the compute nodes you need to ensure you are working on the `/work` filesystem (i.e. the application you want to run is on the work filesystem, and you are running from a directory there), and provide some more filesystem directories to the container at runtime.

This is an example of the command that would be required to run a Singularity container using MPI on the compute nodes on ARCHER2 (this example runs the container interactively using Slurm):

```
export OMP_NUM_THREADS=8
export
SINGULARITYENV_LD_LIBRARY_PATH=/opt/cray/pe/mpich/8.1.23/ofi/gnu/9.1/lib-
abi-
mpich:/opt/cray/pe/pmi/6.1.8/lib:/opt/cray/libfabric/1.12.1.2.2.0.0/lib64:/
usr/lib64/host:/usr/lib/x86_64-linux-
gnu/libibverbs:/.singularity.d/libs:/opt/cray/pe/gcc-libs
export
SINGULARITY_BIND="/opt/cray,/usr/lib64/libibverbs.so.1,/usr/lib64/librdmacm
.so.1, /usr/lib64/libnl-3.so.200,/usr/lib64/libnl-route-
3.so.200,/usr/lib64/libpals.so.0,
/var/spool/slurmd/mpi_cray_shasta,/usr/lib64/libibverbs/libmlx5-
rdmav25.so,/etc/libibverbs.d,/opt/gcc,/work/ta144/ta144/$USER:/workdir"
srun --hint=nomultithread --distribution=block:block --nodes=2 --tasks-per-
node=16 --cpus-per-task=8 --exclusive --partition=standard --qos=short --
reservation=shortqos --account=z19 --time=0:20:0 /usr/bin/singularity run
/work/y07/shared/singularity-images/mpich_base.sif /workdir/mycode
```

The `export SINGULARITYENV_LD_LIBRARY_PATH=…` command sets the libraries that need to be added to the Container when it is started. `export SINGULARITY_BIND=` sets the bind paths for the container to add in the requisite extra filesystems/directories we need to ensure all the network/MPI libraries are available inside the container. This could also be done using the `-B …` flag to the Singularity executable

(`SINGULARITY_BIND` allows you to add bind paths using an environment variable). Then the srun command starts a job on the compute node using 2 nodes, with 16 workers per node, each with 8 threads.

The same commands could be added to a batch script and submitted to the batch system if interactive use was not required. Note, in the bind paths we have added our work directory (`/work/ta144/ta144/$USER:/workdir`) and renamed it `/workdir`. If your directory is at a different location you will need to adjust the bind path definition. Likewise, if you're trying to run an application with a different executable name (something other than `mycode`) this will need to be altered to ensure your specific application runs.

With the above examples it should be possible to compile and run arbitrary MPI code using Singularity on ARCHER2.